

MDA-Multi MICOM USB

(AT89S51/AVR/PIC)

User Guide

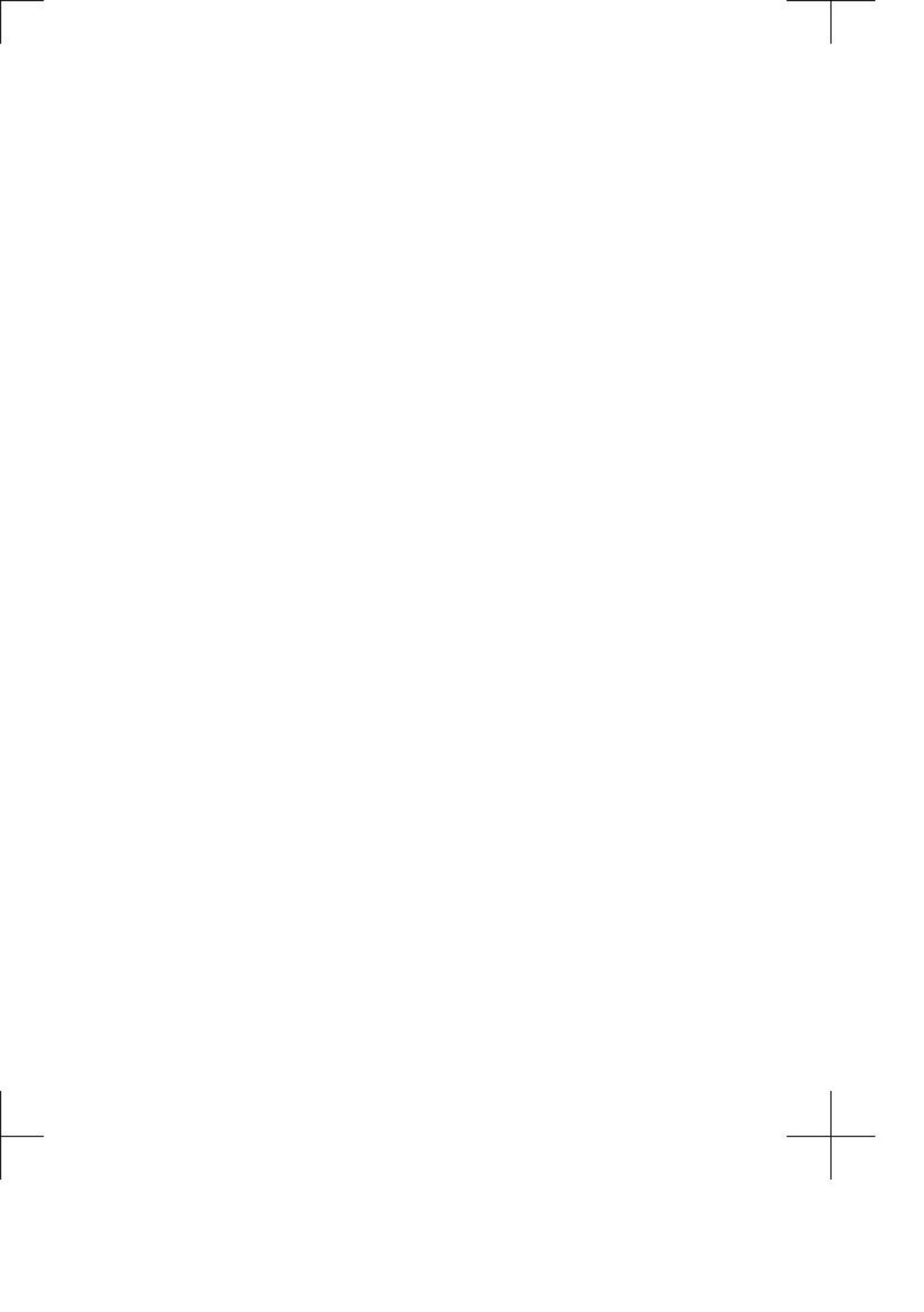


Midas
Engineering
Co., Ltd.

Midas Engineering Co., Ltd.
서울시 구로구 구로동 에이스 비크노타워 5차 906호
Tel : 02-2109-5964, Fax : 02-2109-5963
www.midaseng.com E-mail : midas1@midaseng.com

차 례

1부 MDA-Multi MICOM USB 소개	6
제 1장 MDA-Multi MICOM USB 내용을 확인	8
제 2장 MDA-Multi MICOM USB 설치	9
제 3장 MDA-Multi MICOM USB 소프트웨어 소개	18
제 4장 MDA-Multi MICOM USB 하드웨어 소개	25
2부 MDA-Multi MICOM USB CPU	30
제 1장 AT89S51	32
제 2장 AVR : AT(MEGA)8535	42
제 3장 PIC : PIC16F874/7	54
3부 MDA-Multi MICOM USB 예제	68
제 1장 I/O 포트 연습	70
1. LED 점등하기	70
2. 교통 신호등 만들기	79
3. 7세그먼트 동작시키기	84
4. 7세그먼트 글자 디스플레이하기	90
5. DIP 토탈리 스위치 값 7세그먼트에 디스플레이하기	93
6. 키보드 스캔하기	98
부록 MDA-Multi MICOM USB 회로도	106
1. AT89S51 회로도	108
2. AVR : AT(MEGA)8535 회로도	109
3. PIC : PIC16F874/7 회로도	110
4. MDA-Multi MICOM USB I/O 회로도	111
5. MDA-Multi MICOM USB I/O UP BOARD 회로도	114



주의사항

이 설명서의 저작권은 (주)마이다스엔지니어링에 있습니다.

이 설명서의 일부 혹은 전부를 (주)마이다스엔지니어링의 허가 없이 전자적 기계적 유통적인 어떤 수단으로도 재생산하거나 전송 할 수 없습니다.

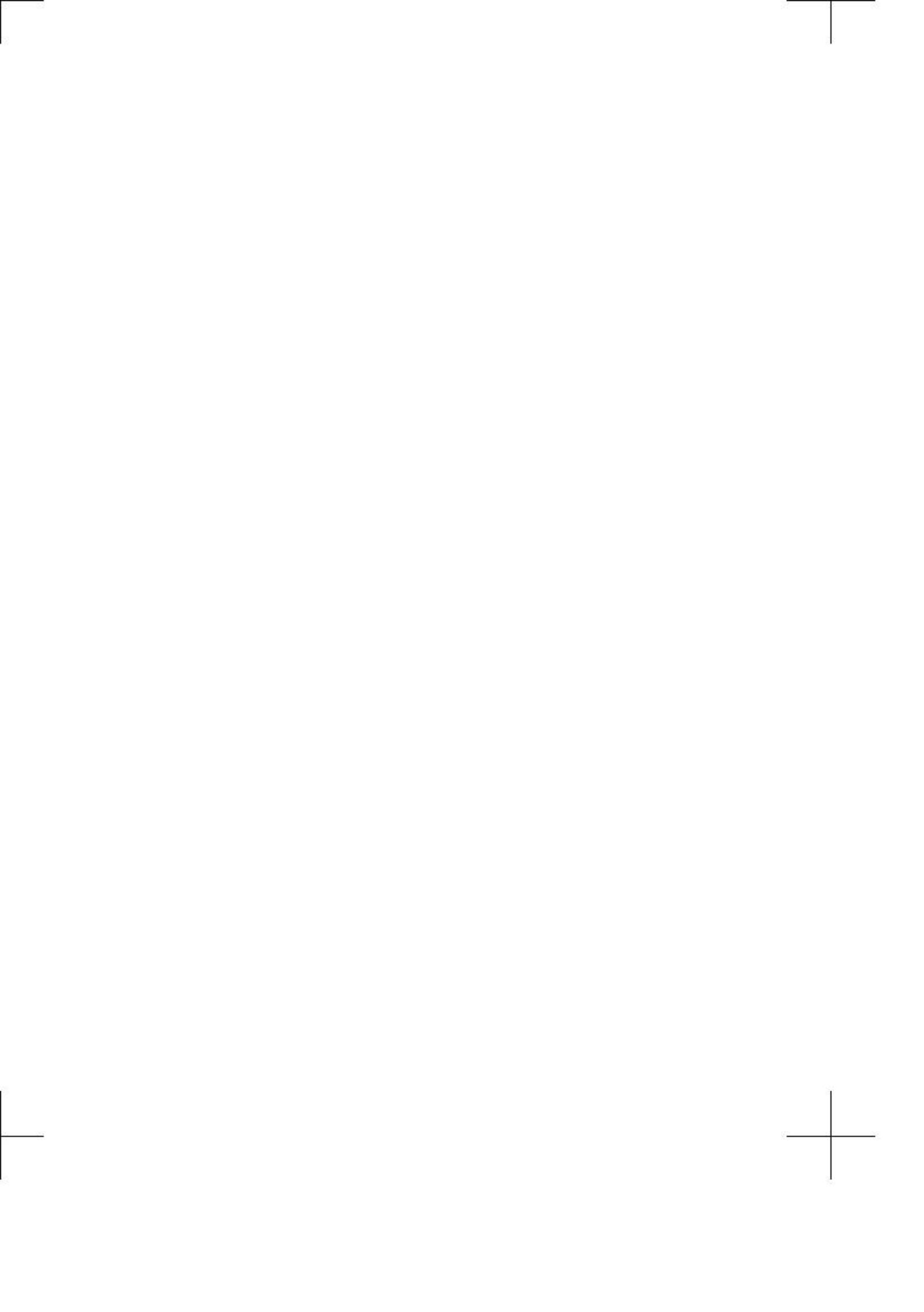
이 설명서의 내용은 제품의 기능 향상 등을 이유로 변경될 수 있습니다.

본 제품을 무단으로 파기하였거나 손상하였을 경우에 (주)마이다스엔지니어링에서는 책임을 지지 않습니다.

본 제품을 손상시키지 않는 범위에서 사용자 영역에 회로를 추가하였을 경우에는 (주)마이다스엔지니어링에 수리를 요청할 때 모두 제거한 다음 보내 주십시오.

고객 지원 서비스

- E-mail : midas1@midaseng.com
- 홈페이지 : www.midaseng.com
- 전화번호는 02-2109-5964이며, fax는 02-2109-5968입니다.
- 본 제품의 수리가 필요하시면 02-2109-5964로 연락하여 (주)마이다스엔지니어링의 안내를 받아 주십시오





1부 MDA-Multi MICOM USB 소개

MDA-Multi MICOM USB 보드는 마이다스 엔지니어링에서 개발된 MICROPROCESSOR 학습용 보드로 **8051, AVR, PIC**을 직접 접속하여 순차적으로 실험해 볼 수 있습니다.

- 제 1장 MDA-Multi MICOM USB 내용물 확인
- 제 2장 MDA-Multi MICOM USB 설치하기
- 제 3장 MDA-Multi MICOM USB 소프트웨어 소개
- 제 4장 MDA-Multi MICOM USB 하드웨어 소개

제 1장 MDA-Multi MICOM USB 내용물 확인

MDA-Multi MICOM USB에서는 아래와 같은 내용물을 제공합니다. 확인 시 문제가 있으면 구입한 곳((주)마이다스 엔지니어링))으로 문의하여 주시기 바랍니다.



☆ MDA-MULTI



☆ 사용자 설명서



☆ MDA-Multi MICOM
USB Studio CD



☆ 전원 연결선



☆ USB 케이블

제 2장 MDA-Multi MICOM USB 설치

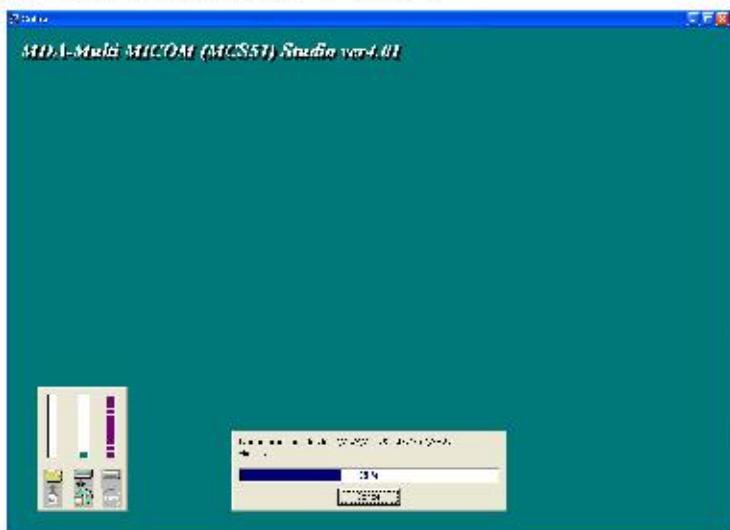
□ Multi MICOM USB Studio 설치방법

1. 귀사가 제공하는 MDA-MULTI USB 설치용 CD를 가동하여 CD 드라이브\MC51 폴더에 있는 SETUP.EXE를 실행시킵니다.

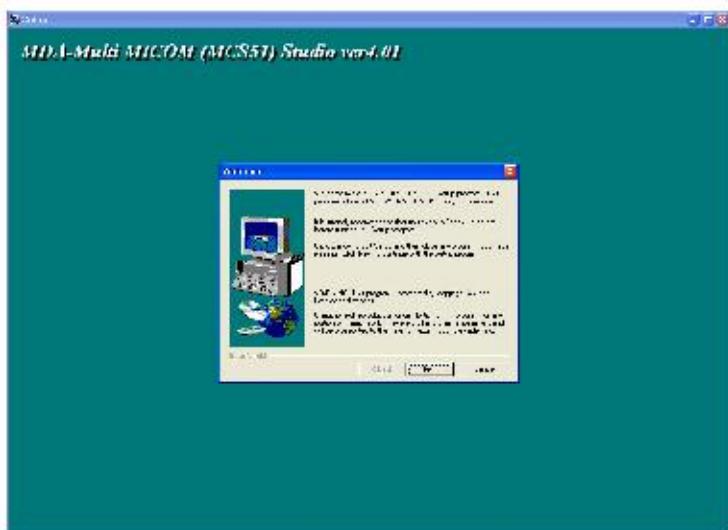


□ 설치화면

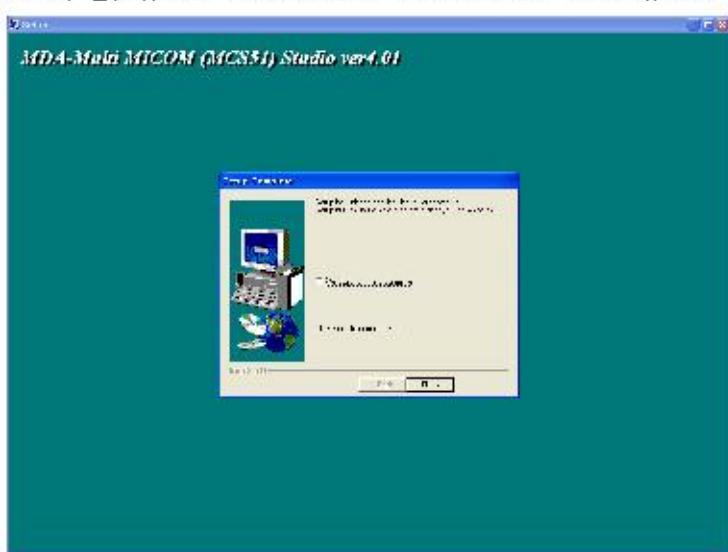
2. MDA-Multi MICOM Studio 화면이 나타납니다.



3. 화면 요구에 따라 NEXT 버튼을 눌러 주시면 다음 화면으로 넘어갑니다.



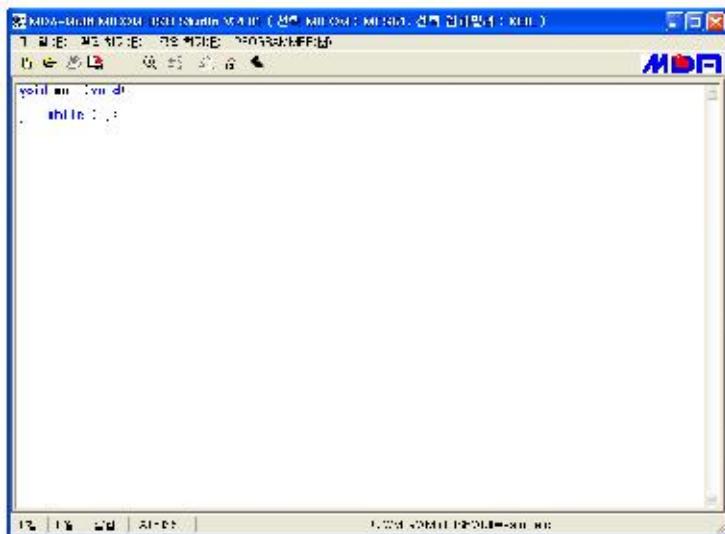
4. 설치가 간단하게 끝났습니다. Finish 버튼을 눌러서 설치를 마무리합니다.



5. 같은 방법으로 MDA-MULTI USB 설치용 CD의 CD 드라이브\AVR\SETUP.EXE 및
CD 드라이브\PIC\SETUP.EXE를 실행시킵니다.
6. 설치가 완료 되었으면 시작→모든 프로그램→MIDASENG→MDA_MULTI_USB을 실
행합니다.

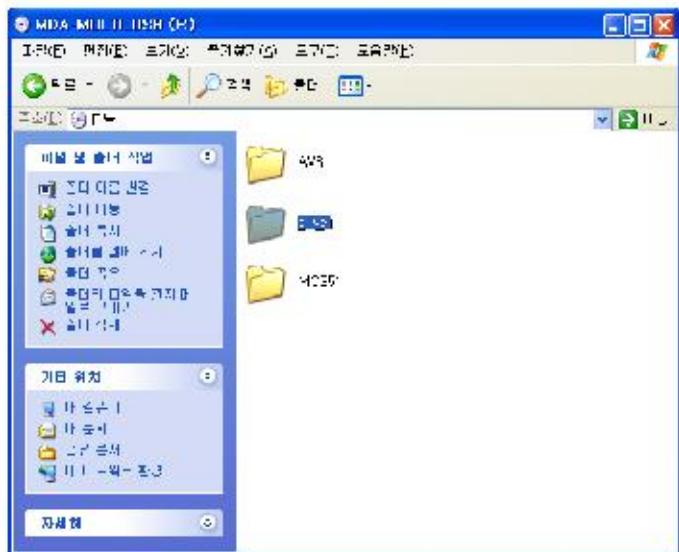
1부/제 2장 MDA-Multi MICOM USB 설치

7. 모니터에 아래 화면이 나타나면 제대로 설치가 된 것입니다.

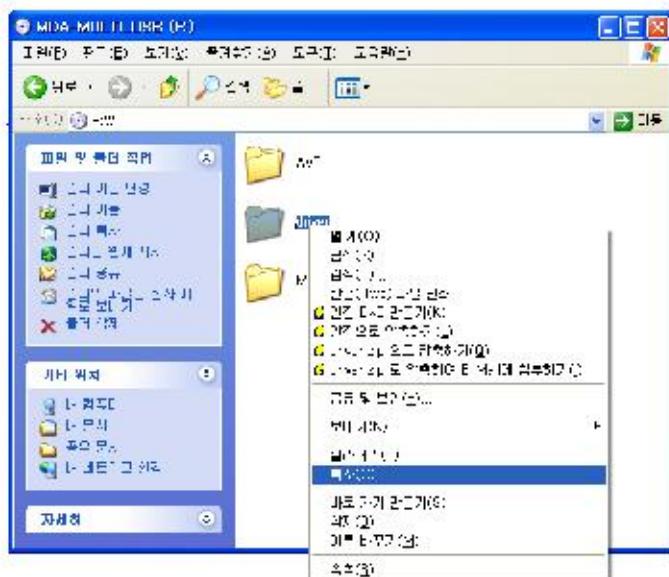


□ Multi MICOM USB driver 설치방법

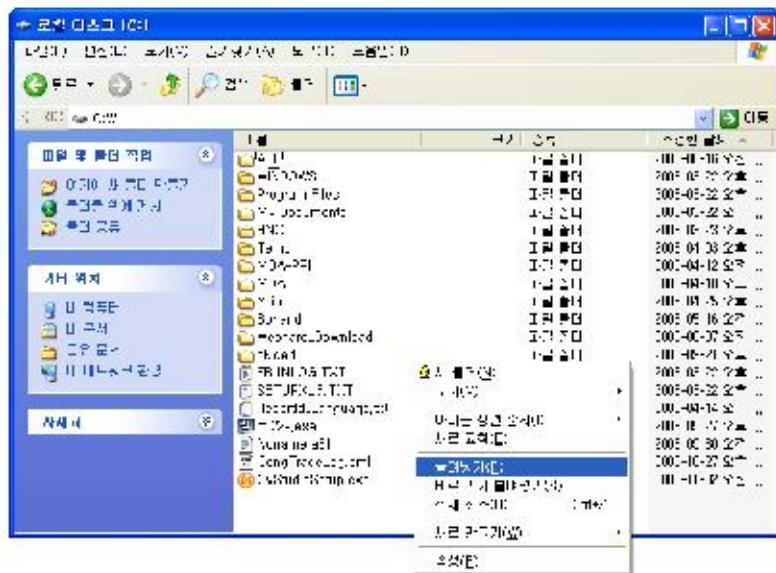
1. MDA-MULTI USB 설치용 CD의 CD 드라이브\driver 폴더를 확인합니다.



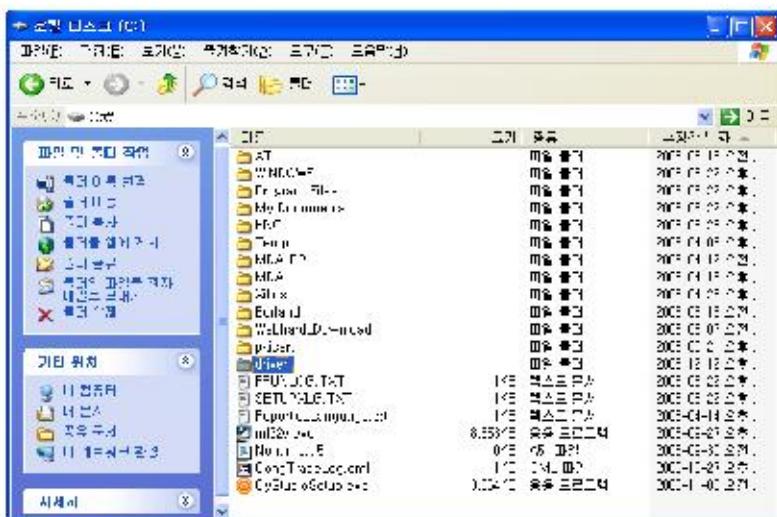
2. CD 드라이브\driver 폴더를 복사합니다.



3. CD 드라이브\driver 폴더를 사용자 하드 디스크에 이동 복사합니다.

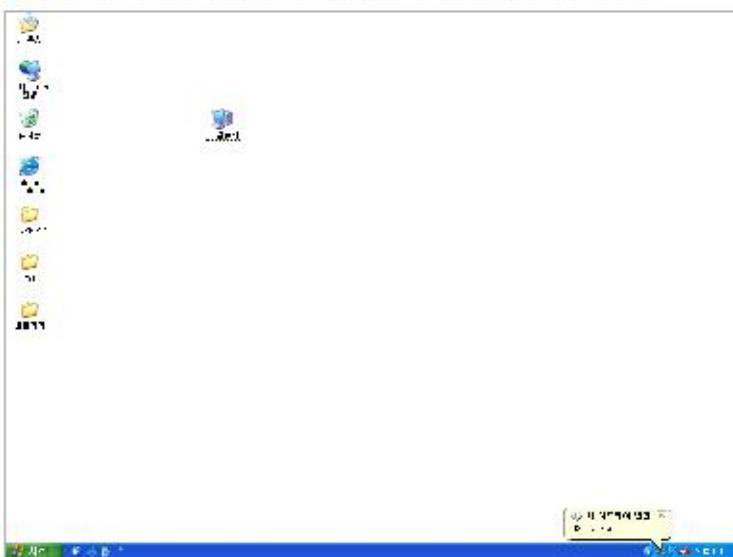


4. 정상적으로 하드디스크에 복사 되었을 때 화면입니다.

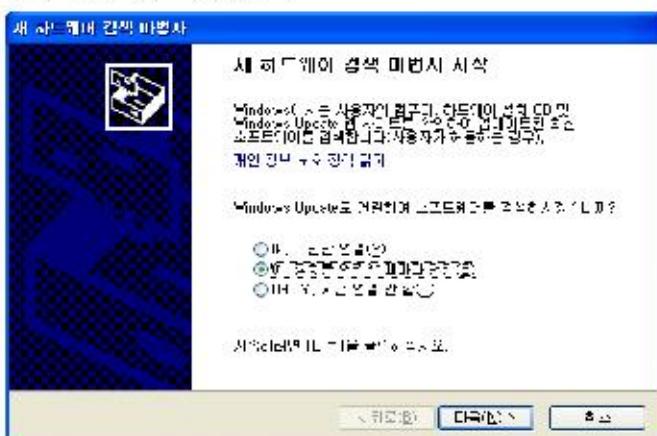


□ Multi MICOM USB driver 설치 방법

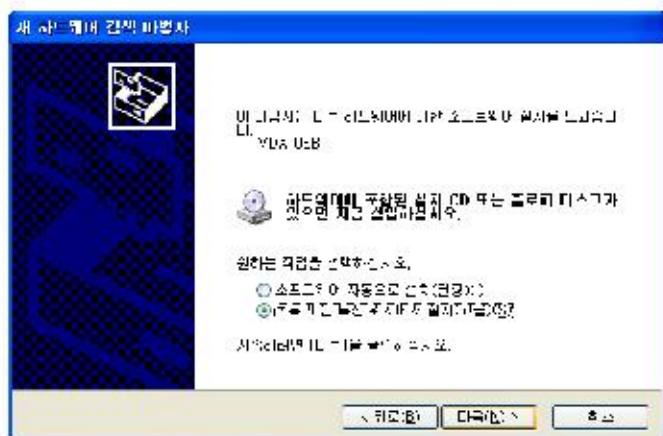
1. MDA-Multi MICOM USB Kit 뒷면 USB포트와 PC를 USB 케이블을 이용하여 연결합니다.
2. Windows-XP에서는 아래와 같이 하드웨어를 자동으로 발견합니다.



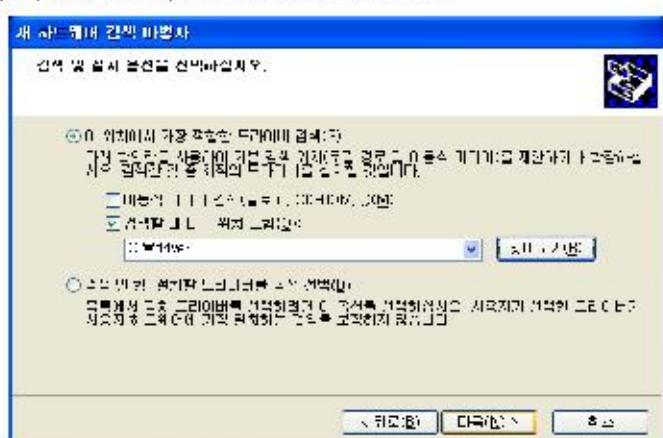
3. 새 하드웨어 검색 마법사가 시작됩니다.



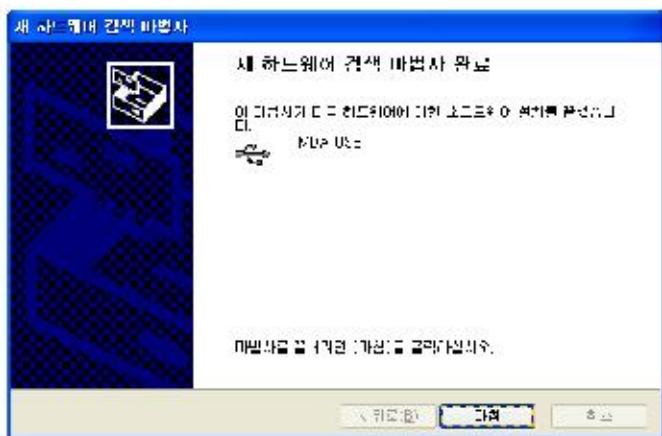
4 고급 설치를 체크 후 다음 버튼을 클릭합니다.



5 하드디스크에 저장된 드라이버 위치를 지정해줍니다.

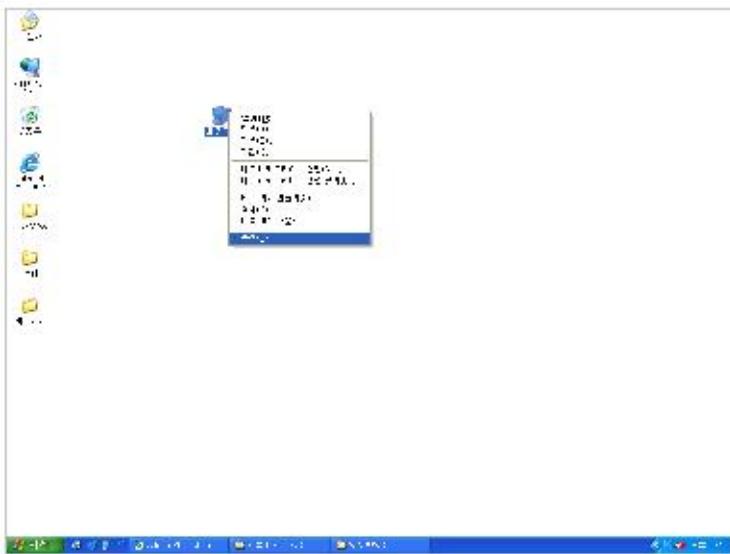


6. MDA-USB 드라이버 설치가 완료 되었습니다.

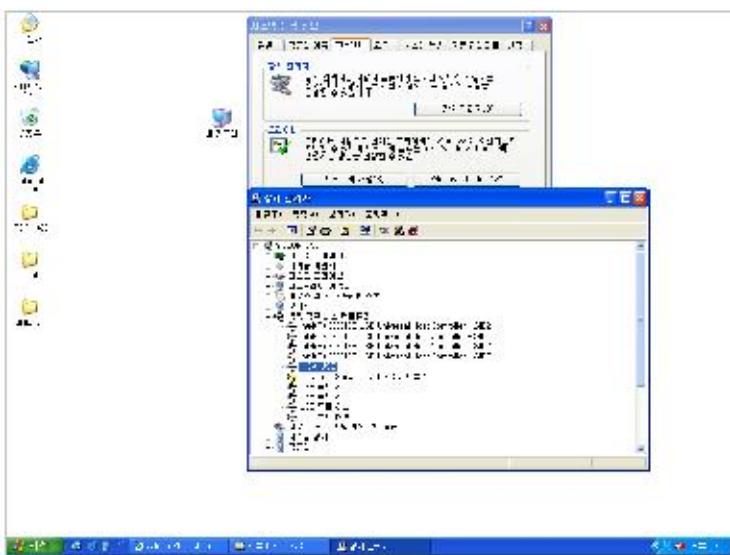


□ Multi MICOM USB driver 설치 확인하기

1. ‘내 컴퓨터’ 아이콘을 마우스 오른쪽 버튼으로 누르고 ‘속성’을 클릭합니다.



2. '하드웨어' 탭에서 장치 관리자 버튼을 클릭하여 MDA-USB 드라이버 설치를 확인합니다.

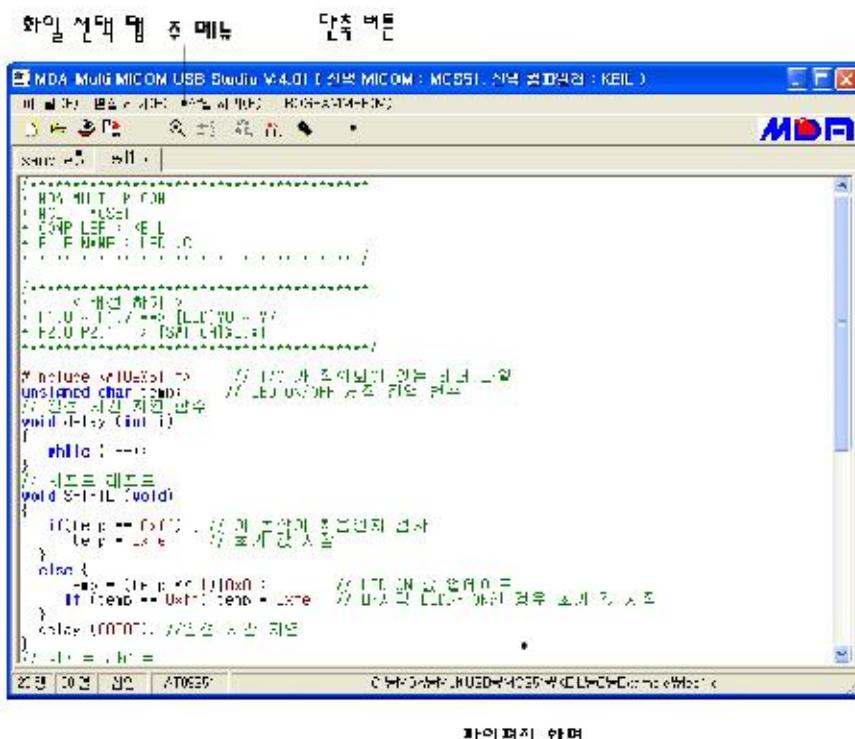


제 3장 MDA-Multi MICOM USB 소프트웨어 소개

여기서는 MDA-Multi MICOM USB을 사용하기 위한 MDA-Multi MICOM USB Studio의 사용법에 대하여 설명하기로 합니다.

MDA-Multi MICOM USB Studio는 소스 파일을 만들어서 어셈블/컴파일을 한 화면에서 실행시킬 수 있으며, 더 나아가서 MDA-Multi MICOM USB에 실행파일을 write할 수 있는 프로그램입니다.

1. MDA-Multi MICOM USB Studio화면의 구성



파일편집 화면

1 주 메뉴

(1) 파일

파일 메뉴를 실행하면 아래와 같이 부 메뉴가 디스플레이 되며, 또 부 메뉴를 실행하면 부 메뉴에 해당되는 화면이 디스플레이 된다.



- 부 메뉴 옆에 있는 Ctrl+N, Ctrl+O...등은 핫키를 말하며
 - 퍼스널 컴퓨터의 키보드에서 Ctrl키를 누른 상태에서 영문자 키를 누르면 바로 해당되는 부 메뉴가 실행된다는 것을 의미 한다.
 - +표시는 Ctrl 키를 누른 상태에서 다른 키를 누른다는 것을 의미 한다.

(2) 평집하기

편집하기 메뉴를 실행하면 아래와 같이 부 메뉴가 디스플레이 되며, 또 부 메뉴를 실행하면 부 메뉴에 해당되는 화면이 디스플레이 된다. 이 메뉴의 사용법은 일반적인 윈도우에서 사용하는 에디터 사용 방법과 같다.



- 부 메뉴 옆에 있는 Ctrl+A, Ctrl+X,..등은 핫키를 말하며
 - 퍼스널 컴퓨터의 키보드에서 Ctrl키를 누른 상태에서 영문자 키를 누르면 바로 해당되는 부 메뉴가 실행된다는 것을 의미 한다.
 - +표시는 Ctrl 키를 누른 상태에서 다른 키를 누른다는 것을 의미 한다.

(3) 작업하기

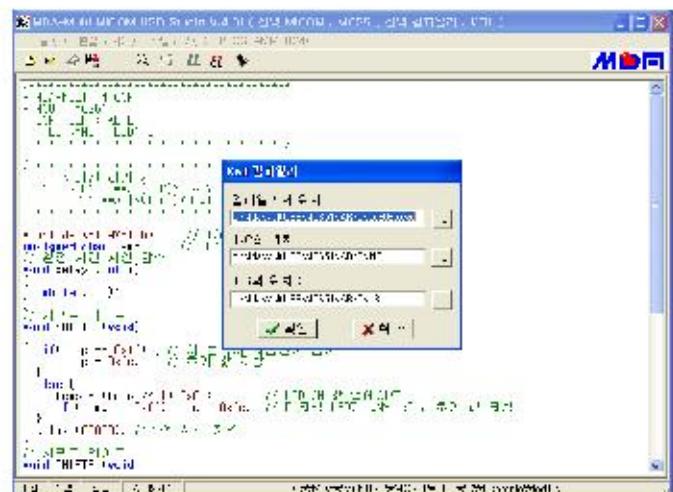
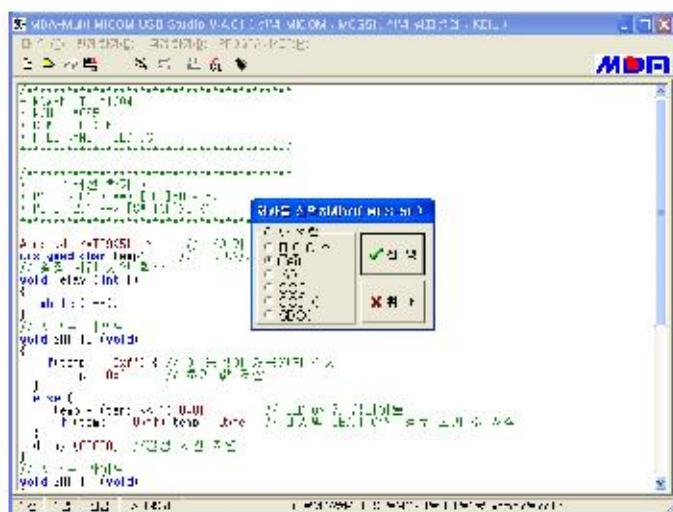
편집하기 메뉴를 실행하면 아래와 같이 부 메뉴가 디스플레이 되며, 또 부 메뉴를 실행하면 부 메뉴에 해당되는 화면이 디스플레이 된다.



- 어셈블과 컴파일은 동시에 액티브 되지 않고 소스 파일에 의해서 자동으로 설정된다



- 컴파일러/어셈블러 설정하기 : 사용자 컴파일러 혹은 어셈블러를 선택하기 위한 작업이다.



- 마이컴 선택하기



사용자는 마이컴을 선택하여 사용자 프로그램을 작성한다.

2 단축 버튼

MDA-Multi MICOM USB Studio에서는 사용자가 주 메뉴와 부 메뉴를 선택 해서 실행하는 번거로움을 덜기 위해서 단축 버튼을 만들어 놓았다.



단축 버튼	단축 버튼 이름	주 메뉴	부 메뉴
	새 파일	파일	새 파일
	파일 오픈	파일	파일 오픈
	파일 닫기	파일	파일 닫기
	파일 저장	파일	파일 저장
	찾기	편집하기	찾기
	실행 취소	편집하기	실행 취소
	어셈블 & 링크	작업하기	어셈블 & 링크
	컴파일 & 링크	작업하기	컴파일 & 링크
	마이콤 라이트	PROGRAMMER	

3 파일 선택 탭



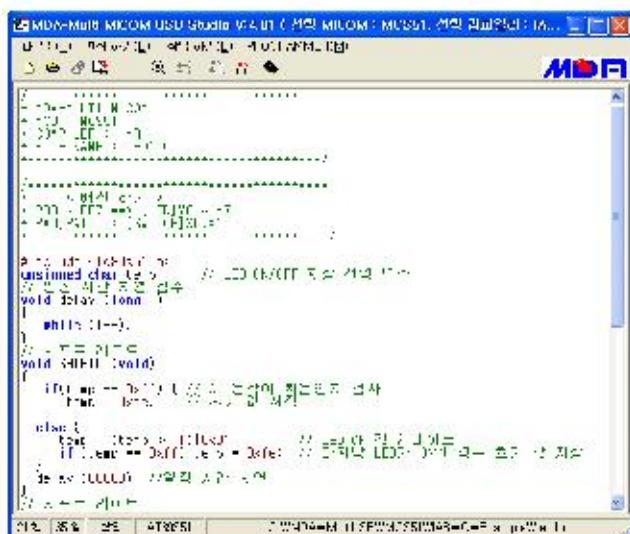
여러 소스 파일을 선택 할 때 사용한다.

2. MDA-Multi MICOM USB Studio 사용하기

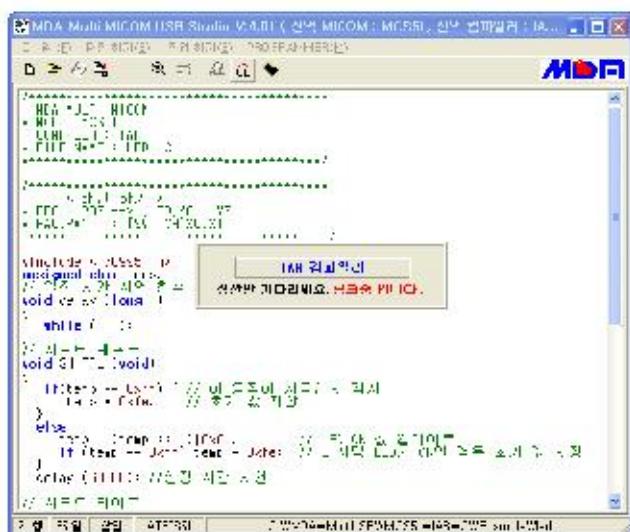
MDA-Multi MICOM USB Studio 실행



2 소스 파일 작성

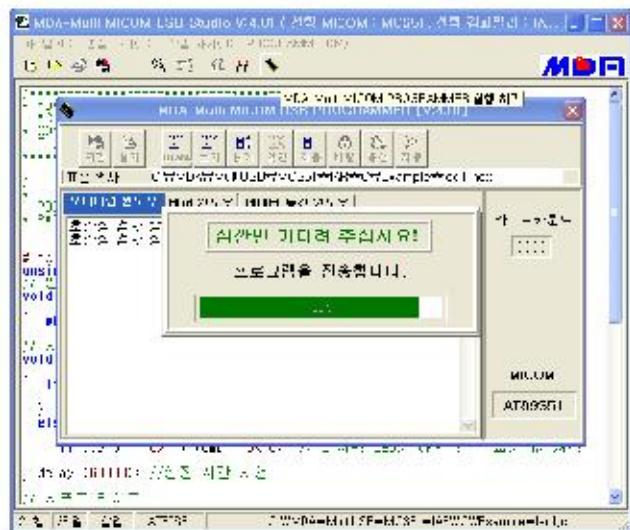


3 커뮤니케이션 파일/링크

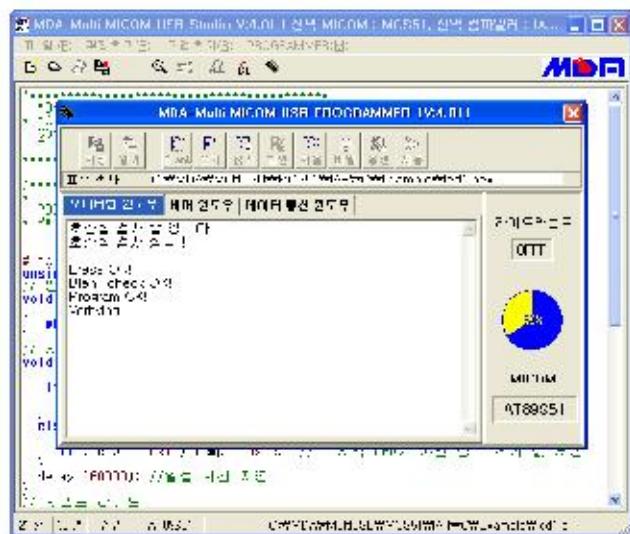


- 소스 파일을 기계어/HEX 파일로 만든다.

4 PROGRAMMER 실행

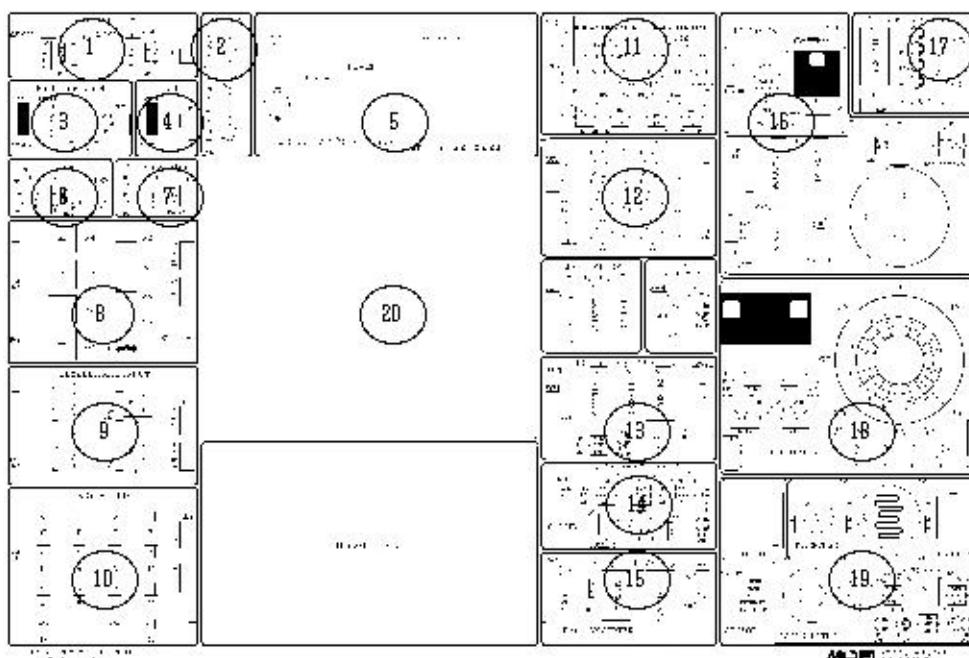


5 핵사 파일 WRITE



제 4장 MDA-Multi MICOM USB 하드웨어 소개

아래 그림의 MDA-Multi MICOM USB는 모든 실험을 사용자가 CPU 보드와 직접 선으로 연결하여 실험하도록 구성되어 있습니다.



1 Serial EEPROM

SPI모드와 I²C모드를 실험하기 위한 EEPROM 93LC066과 24LC32가 연결되어 있으며, 사용자는 각각 CPU보드와 직접 선을 연결하여 실험 합니다.

2 POWER ON/OFF 스위치

MDA-Multi MICOM USB 보드의 전체 전원을 ON/OFF하는 스위치이다. ON 하였을 경우 +5V와 +12V LED가 ON 됩니다.

3 REAL TIME CLOCK

디지털 시계로 많이 사용되고 있는 DS1302를 CPU보드와 직접 선을 연결하여 실험 합니다.

4 CLOCK

7segment를 이용하여 타이머를 실험 합니다.

5 LCD DISPLAY (16 × 2 LINE)

16 × 2 LINE 캐릭터 LCD를 사용하며, 사용자는 CPU 보드와 직접 선을 연결하여 실험 합니다.

6 TIMER

타이머모드 실험을 하기위한 NE555가 연결되어 있으며, 사용자는 CPU보드와 직접 선을 연결하여 실험 합니다.

7 TEMPERATURE

디지털 온도계로 많이 사용되고 있는 DS1620가 연결되어 있으며, 사용자는 CPU보드와 직접 선을 연결하여 실험 합니다.

8 SWITCH

TACT 스위치와 토글 스위치가 각각 연결되어 있으며, 사용자는 CPU보드와 직접 선을 연결하여 실험 합니다.

9 HEXA DECIMAL INPUT

DIP 로터리 스위치가 각각 연결되어 있으며, 사용자는 CPU보드와 직접 선을 연결하여 실험 합니다.

10 KEY MATRIX

4 × 4 키가 행 • 열로 연결되어 있으며, 사용자는 CPU보드와 직접 선을 연결하여 실험 합니다.

11 7SEGMENT

1부/제 4장 MDA-Multi MICOM USB 하드웨어 소개

콤온 애노드 (Common Anode)형 2-DIGIT 7SEGMENT가 연결되어 있으며, 사용자는 CPU보드와 직접 선을 연결하여 실험 합니다.

[12] LED

10Ø LED가 연결되어 있으며, 사용자는 CPU보드와 직접 선을 연결하여 실험 합니다.

[13] USRT

송수신 실험을 하기 위한 회로가 연결 되어 있으며, 사용자는 CPU보드와 직접 선을 연결하여 실험합니다.

[14] SOUND

SOUND 실험을 하기 위한 회로가 연결되어 있으며, 사용자는 CPU보드와 직접 선을 연결하여 실험 합니다.

[15] D/A CONVERTER

AD7302를 이용하여 디지털 신호를 아날로그 신호로 변환하여 D.C 모터를 구동할 수 있도록 되어 있습니다.

[16] DC MOTOR

DC MOTOR 실험을 하기 위한 회로가 연결되어 있으며, 사용자는 CPU보드와 직접 선을 연결하여 실험 합니다.

[17] RS-232C

데이터 통신의 여러 기능 실험을 하기 위한 RS-232 회로가 연결되어 있으며, 사용자는 CPU보드와 직접 선을 연결하여 실험 합니다.

[18] STEP MOTOR

STEP MOTOR 실험을 하기 위한 회로가 연결되어 있으며, 사용자는 CPU보드와 직접 선을 연결하여 실험 합니다.

[19] SENSOR

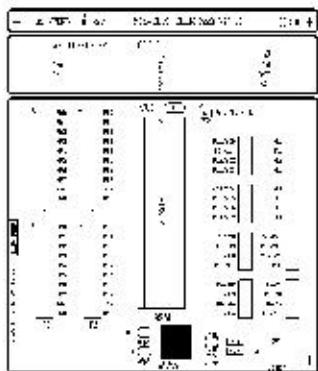
SENSOR 실험을 하기 위한 회로가 연결되어 있으며, 사용자는 CPU보드와 직접 선을 연결

하여 실험 합니다.

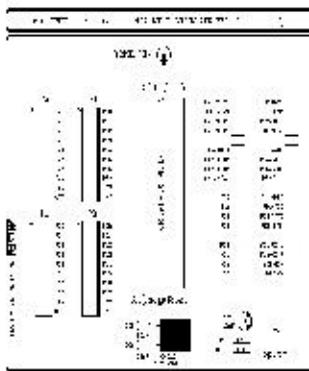
20 MDA-Multi MICOM USB CPU 인터페이스 보드

사용자가 임의로 MDA-Multi MICOM USB 응용장치에 연결하여 사용할 수 있도록 왼쪽 컨넥터가 연결되어 있고, 오른쪽 컨넥터는 사용자 확장 보드에 연결하여 사용할 수 있도록 컨넥터를 연결하여 놓았습니다.

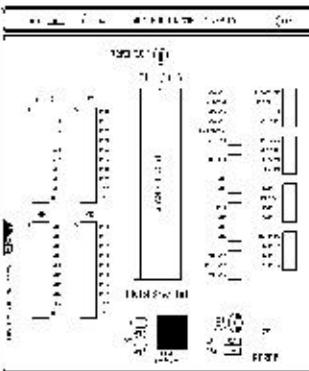
이 각각의 MDA-Multi MICOM USB CPU 인터페이스 보드는 MDA-Multi MICOM I/O 보드 없이 독립적으로 사용이 가능합니다.



89S51



AT(MBGA)89S51



PIC16F877



2부 MDA-Multi MICOM USB CPU

MDA-Multi MICOM USB 보드는 AT89S51, AVR, PIC를 기본적으로 실험, 실습할 수 있도록 제작하였습니다.
사용자는 각 CPU를 교체하여 작업합니다.

- 제 1장 AT89S51
- 제 2장 AVR : AT(MEGA)8535
- 제 3장 PIC : PIC16F874/7

제 1장 AT89S51

AT89S51은 4K 바이트의 플래시(flash)메모리, 128바이트의 내부 RAM, 4개의 I/O 포트, 2개의 16비트 타이머/카운터, 5개의 인터럽트 소스, 전 이중의 시리얼 포트, 워치독 타이머 등을 갖고 있다.

1. AT89S51의 특징

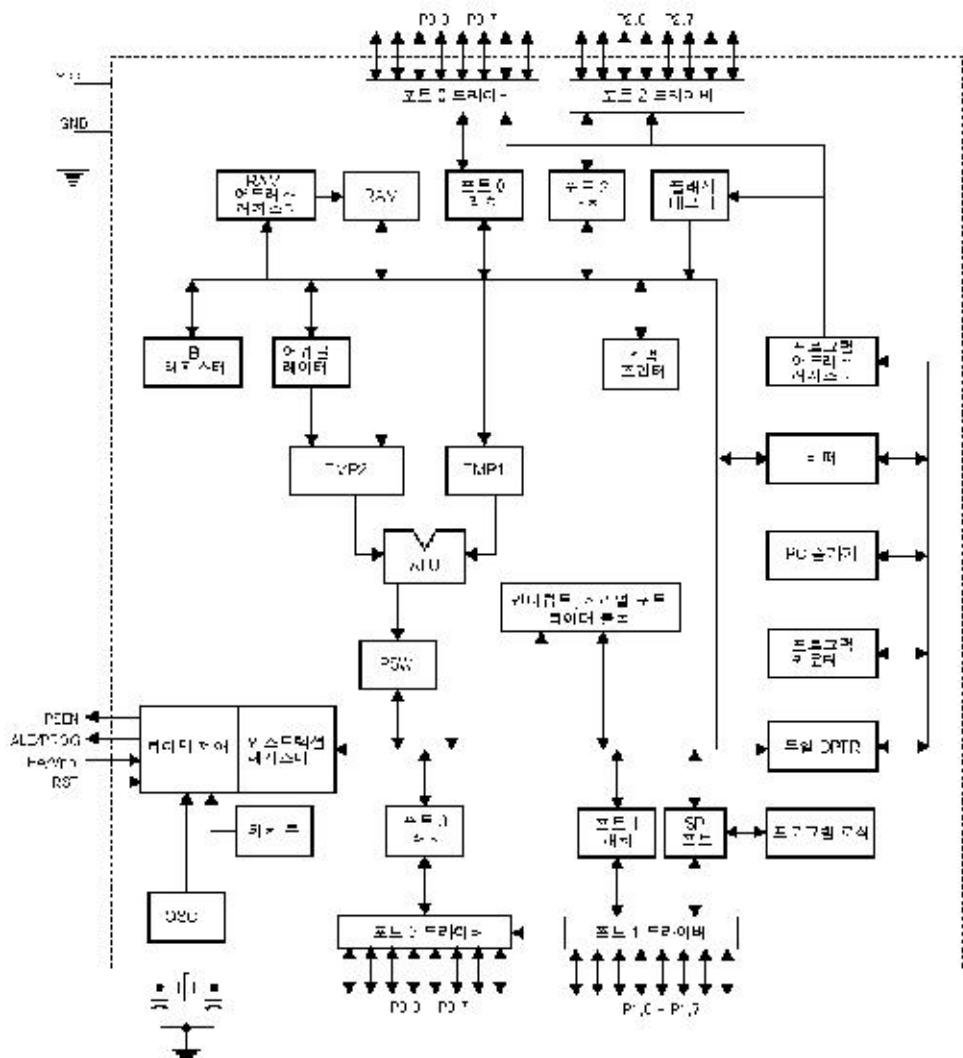
AT89S51의 특징을 열거하면 다음과 같다.

- 8051 마이컴과 호환
- ISP(In System Programmable) 방법의 4K바이트 플래시(flash) 메모리를 1000번 쓰기/지우기 가능
- 4.0[V]~5.5[V]에서 동작, 클록은 0~33[MHz]에서 동작 가능
- 3단 프로그램 메모리 록(lock) 가능
- 128 × 8 내부 RAM
- 32비트의 I/O 및 2개의 16비트 타이머/카운터, 6개의 인터럽트 소스, 전 이중(full duplex)방법의 시리얼 채널
- 저 전력 아이들 및 인터럽트에서 벗어날 수 있는 파워 다운 모드
- 워치독 타이머, 2개의 데이터 포인터(DPTR), 파워 오프 플래그

2. AT89S51의 내부 기능

프로그램 카운터(PC : Program Counter)

프로그램 카운터는 프로그램 메모리에 저장되어 있는 명령의 순서를 정하는 16비트 레지스터로서 실행시킬 명령이 기억되어 있는 프로그램 메모리의 번지를 가리키고 있다. 프로그램 카운터는 리셋되면 0x0000으로 된다.



AT89S51의 내부 블록도

2 PSW(Program Status Word)

PSW 레지스터는 아래의 플래그로 구성되어 있다.

비트번호	7	6	5	4	3	2	1	0
플래그명칭	CY	AC	F0	RSI	RS0	OV	-	P
⇒	캐리	보조캐리	사용자	뱅크선택 비트	오버플로		파리티	

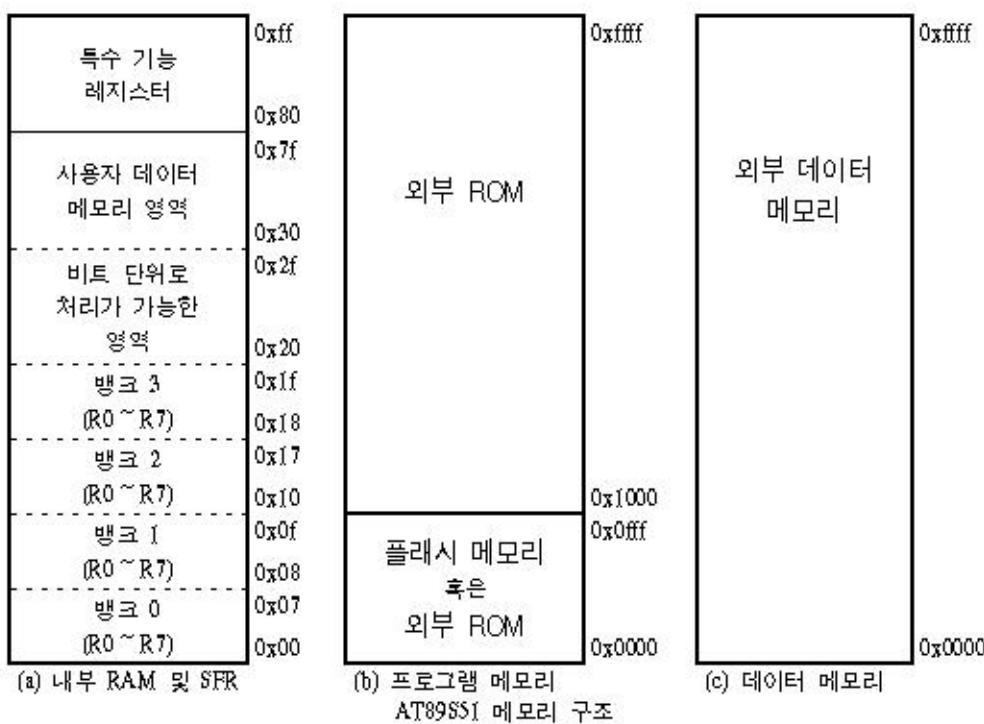
3 SP(Stack Pointer)

이 레지스터는 8비트 크기로 스택의 시작번지를 가리키는데 사용하며, 스택을 반드시 내부 데이터 메모리를 사용하여야 한다. SP는 리셋되면 0x07로 설정된다.

3. AT89S51의 메모리 구조

AT89S51의 메모리 구조는 아래 그림과 같으며, 각 메모리의 기능은 다음과 같다.

- 내장된 내부 데이터 메모리 128바이트(어드레스 0x00~0x7f)
- 내장된 내부 특수기능 레지스터 128바이트(어드레스 0x80~0xff)
- 내장된 프로그램 메모리 4K바이트(어드레스 0x0000~0xffff)
- 외부 확장 프로그램 메모리 60K/64K바이트(어드레스 0x1000/0x0000~0xffff)
- 외부 확장 데이터 메모리 64K바이트(어드레스 0x0000~0xffff)



1 내부 데이터 메모리

AT89S51은 내부 데이터 메모리로 128바이트(어드레스 0x00~0x7f)가 있다. 내부 데이터 메모리는 데이터를 저장 혹은 스택 등의 여러 용도로 사용할 수 있는 메모리이다.

● **레지스터 뱅크 0~3**

어드레스 0x00~0x1f로 할당되어 있으며, 8개씩 묶어서 뱅크(bank)라 부른다.

● **비트 어드레스 영역**

메모리 어드레스 0x20~0x2f의 16바이트(128비트)가 할당되어 있으며 이 영역 트 번호 0x00~0x7f로도 액세스 할 수 있다.

● **사용자가 임의로 사용할 수 있는 데이터 메모리**

나머지 0x30~0x7f의 영역은 사용자가 임의의 데이터 RAM으로 사용할 수 있다.

2 특수 기능 레지스터(SFR : Special Function Register)

AT89S51의 모든 입·출력 기능은 특수 기능 레지스터(SFR : Special Function Register)를 이용하여 상위 어드레스(0x80~0xff)가 할당되어 있다.

0xf8								0xff
0xf0	B 00000000							0xf7
0xe8								0xef
0xe0	ACC 00000000							0xe7
0xd8								0xd7
0xd0	PSW 00000000							0xd7
0xc8								0xcf
0xc0								0xc7
0xb8	IP xx000000							0xbf
0xb0	P3 11111111							0xb7
0xa8	IB 0x000000							0xaf
0xa0	P2 11111111		ANXRI xxxxxx0			WDTRST xxxxxx0		0xa7
0x98	SCON 00000000	SBUF xxxxxxx						0x9f
0x90	P1 11111111							0x97
0x88	TCON 00000000	TMOD 00000000	TLO 00000000	TLI 00000000	TH0 00000000	TH1 00000000	AUXR xx00xx0	0x8f
0x80	P0 11111111	SP 00000111	DPOL 00000000	DPOH 00000000	DPLL 00000000	DPLH 00000000	PCON 0xx0000	0x87

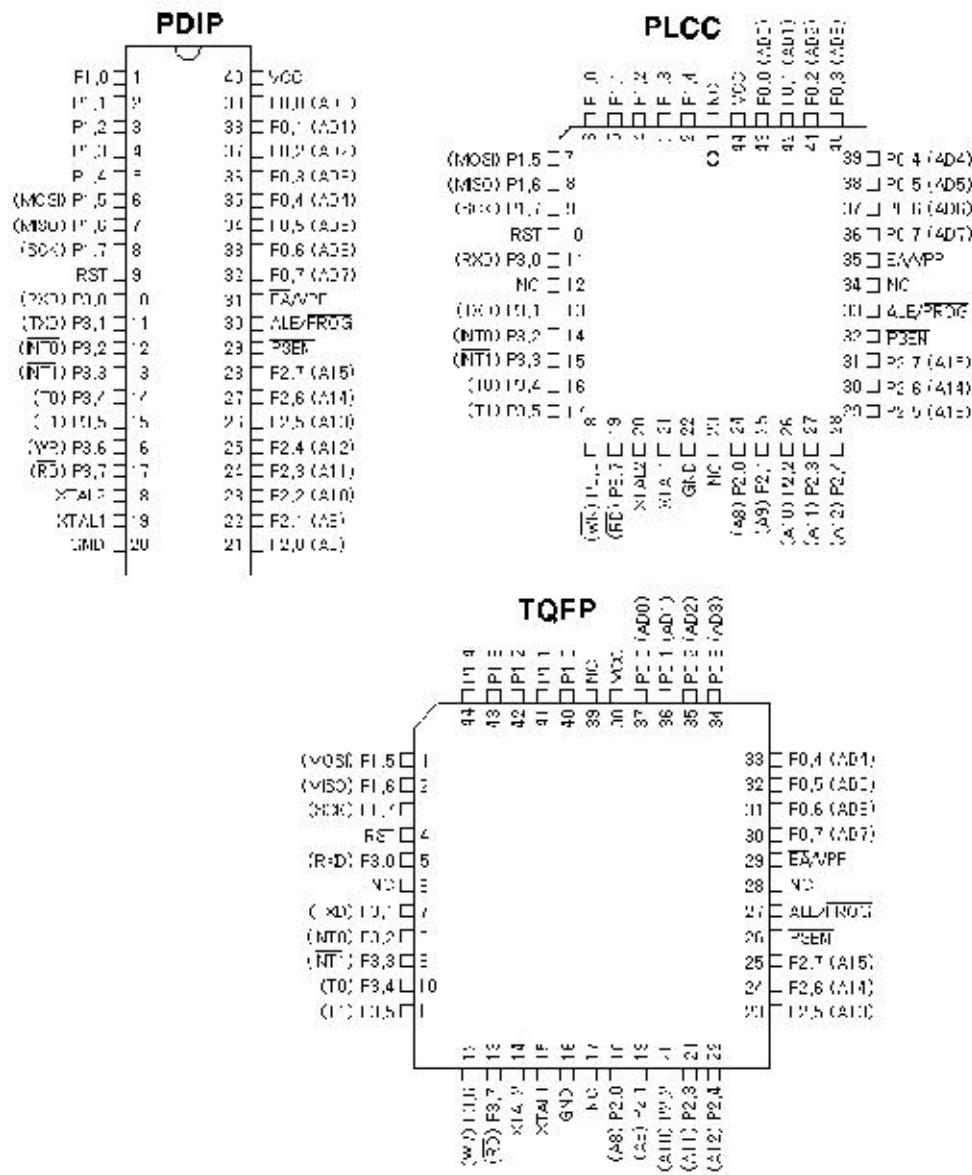
▣ 음영으로 된 부분은 비트 단위 처리가 가능하고, X로 된 부분은 8051에는 없는 부분이다.

▣ 할당되지 않은 부분을 읽기/쓰기하면 올바른 데이터를 보증할 수 없다.

SFR의 이름 및 어드레스, 리셋 값

4. AT89S51의 핀의 기능

AT89S51은 DIP(Dual In line Package), PLCC(Plastic Leaded Chip Carrier), TQFP(Thin Quad Flat Pack)의 형태가 있다.



AT89SS1핀 배치도

1 포트 0 (P1.0~P1.7)

8비트 오픈 드레인(Open Drain) 양 방향성 입·출력 단자이다.

- 외부에 메모리를 인터페이스 하지 않았을 때에는 범용 입·출력 포트로 사용할 수 있다. 포트 0에 "1"을 쓰면 하이 임피던스(high impedance) 입력으로 사용될 수 있다.
- 외부에 프로그램 메모리와 데이터 메모리를 인터페이스 하였을 때, 하위 어드레스 A0~A7과 데이터 버스로 사용된다. 이때 내부적으로 풀업(pull-up)되어 있다.

2 포트 1 (P1.0~P1.7)

내부 풀업(pull-up)이 되어 있는 8비트 양방향성 입·출력 단자이며, 범용 입·출력 포트로 사용된다. 또 P1.5, P1.6, P1.7은 ISP(In System Programmable)방법으로 플래시 메모리로 프로그램을 써 넣을 때 사용된다.

P1.5, P1.6, P1.7의 다른 기능

포트 핀	다른 기능	설 명
P1.5	MOSI(Master Out Serial In)	ISP 프로그래밍에서 데이터 출력
P1.6	MISO(Master In Serial Out)	ISP 프로그래밍에서 데이터 입력
P1.7	SCK(Serial Clock)	ISP 프로그래밍에서 클럭 입력

3 포트 2 (P2.0~P2.7)

내부 풀업(pull-up)을 갖는 8비트 양방향성 입·출력 단자이다.

- 외부에 메모리를 인터페이스하지 않았을 때에는 범용 입·출력 포트로 사용할 수 있다.
- 외부에 메모리를 연결 하였을 경우에는 상위 어드레스 A8~A15로 사용된다.
- 외부에 8비트 어드레스만을 사용하는 메모리를 연결해서 C-언어의 pdata(어셈블리어의 MOVX A,@Ri 또는 MOVX @Ri,A 명령)포인터를 사용할 경우 포트 2에 SFR의 포트 2 내용이 출력된다.

4 포트 3 (P3.0~P3.7)

내부 풀업(pull-up)을 갖는 8비트 양방향성 입·출력 단자이며, 이 포트는 다음과 같이 2개의 기능을 갖고 있다.

- 범용 입·출력 포트로 사용할 수 있다.
- 범용 입·출력 포트의 기능이 아닌 아래 표와 같은 기능으로도 사용할 수 있다.

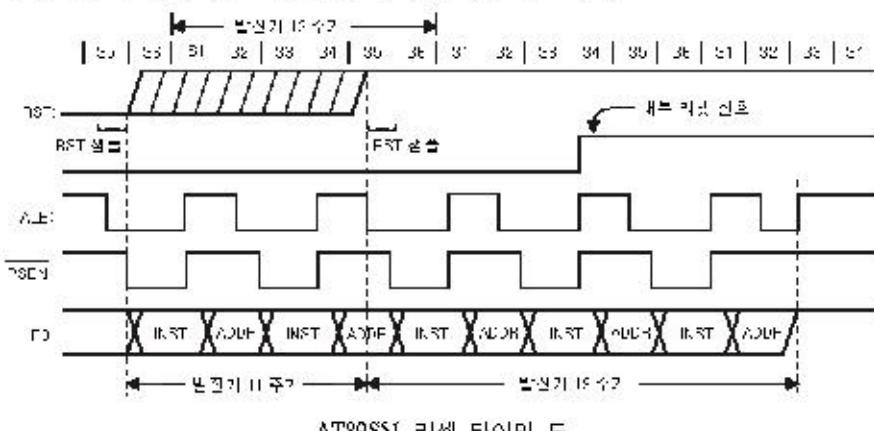
포트 3의 다른 기능

포트 핀	기 능	설 명
P3.0	RxD(receive data)	수신 데이터 입력
P3.1	TxD(transmitte data)	송신 데이터 출력
P3.2	INT0(external interrupt 0)	외부 인터럽트 0 입력
P3.3	INT1(external interrupt 1)	외부 인터럽트 1 입력
P3.4	T0(timer 0 external input)	외부 클록 0 입력
P3.5	T1(timer 1 external input)	외부 클록 1 입력
P3.6	WR(write)	외부 데이터 메모리 라이트 제어신호
P3.7	RD(read)	외부 데이터 메모리 리드 제어신호

▣ 이들 핀 중에서 사용하지 않는 핀은 독립적으로 입·출력 포트의 기능으로 사용할 수 있다.
예를 들면, 데이터 통신 핀을 사용하지 않을 경우에는 P3.0, P3.1로 사용할 수 있다.

5 RESET : 입력, 액티브 H(high)

리셋은 발진기가 동작하고 있는 상태에서, RES핀이 최소한 2개의 머신 사이클(발진기 24 주기) 동안 "H"가 입력되면 리셋된다. 되어 있어야 유효하다.



AT89S51 리셋 타이밍 도

- 외부 리셋 신호는 내부 클록과 비동기적으로 동작하므로, 머신 사이클의 페이스 2-스테이트 5마다 RST핀을 샘플 한다.
- RST핀에서 "H"를 검출한 후 발진기 19개 주기 동안은 현 상태를 유지하게 된다. 즉, 외부 리셋 신호가 RST핀에 공급되고 난 후, 발진기 19~31 주기 동안이 된다.
- RST핀이 "H"동안 ALE, PSEN은 약한 "H" 상태가 되고, RST 핀이 "L"이 된 후, 1 혹은 2 머신 사이클이 지난 후 ALE, PSEN 신호는 동작하게 된다.
- ALE, PSEN핀이 "0"이 되는동안 CPU는 불확실한 상태로 들어가고, 내부 리셋 알고리즘은 포트래치, 스택 포인터, SBUF를 제외하고는 모든 SFR에 "0"을 써 넣는다. 포트 래치는 0xff, 스택포인터는 0x07, SBUF는 확정이 안 된 값으로 초기 설정된다.
- 내부 RAM은 리셋에 영향을 받지 않는다. 파워 ON 리셋에서도 RAM 의 내용은 확정된 값이 없는 상태가 된다.

6 ALE (Address Latch Enable) : 출력, 액티브 H(high)

외부 프로그램 메모리 혹은 데이터 메모리가 인터페이스가 되어있는 경우, 포트 0은 어드레스 버스(A0~A7)겸 데이터 버스(D0~D7)로 사용된다. 이때 ALE 신호는 포트 0이 어드레스 버스 A0~A7로 동작하고 있다는 것을 외부에 알리는 신호이다.

- AUXR 레지스터의 DISABLE이 "0"인 경우, ALE는 보통 동작에서는 발진기 주파수의 1/16의 주파수가 출력하고 있어서, 외부에서 펄스신호로 사용할 수 있다. 이때 외부 메모리를 액세스하는 동안은 펄스가 출력되지 않는다.
- AUXR 레지스터의 DISABLE이 "1"인 경우 외부 데이터 메모리를 액세스 할 때 예만 동작한다.

7 PSEN (Program Strobe enable) : 출력, 액티브 L(low)

외부 프로그램 메모리에서 데이터를 읽기 위한 리드 신호가 된다. 데이터 메모리에서 데이터를 읽을 때에는 동작하지 않는다.

7 EA (External Access enable) : 입력, 액티브 L(low)

외부 프로그램 메모리를 사용할 것인가를 결정한다.

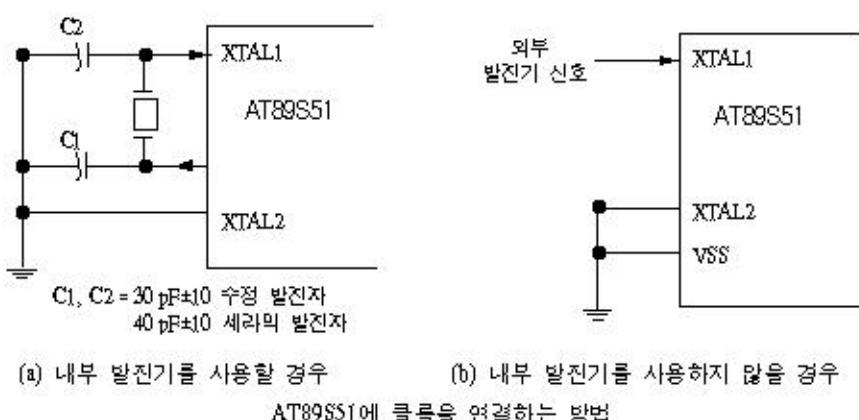
- $\overline{EA}=0$: 0x0000 ~ 0x0ffff (4K 바이트) 전 영역을 외부 ROM을 사용한다.
- $\overline{EA}=1$: 0x0000 ~ 0x0ffff (4K 바이트) 전 영역을 내부 플래시 메모리를 사용한다. 이 경우 0x1000번지 이상을 액세스 하면, 자동으로 외부 ROM이 선택된다.

8 XTAL1 : 입력

인버팅(inverting) 발진기 증폭기의 입력, 내부 클록 동작 회로 입력을 받는다.

9 XTAL2 : 출력

인버팅(inverting) 발진기 증폭기에서 출력한다.



10 Vss, Vcc : 접지, +5V

제 2장 AVR : AT(MEGA)8535

AVR 마이컴은 ATMEL사에서 만든 논리 회로 및 불취발성 메모리가 같이 내장되어 있는 플래시·마이컴(flash·micom)이다. 8비트 RISC(Reduced Instruction Set Computer)구조로 되어 있는 AVR CPU 코더(core)와 프로그램 저장용 플래시·메모리(flash·memory) 및 데이터 저장용 EEPROM을 원-칩(one-chip)화 한 프로세서이다.

일반적으로 범용 원-칩 마이컴은 CMOS형의 프로세서로 만들어진 마스크 ROM으로 되어 있는 것을 의미하고, 일반 사용자는 마스크 ROM을 사용할 수 없고, 외부에 EPROM 등을 인터페이스해서 사용하기 때문에, 결과적으로 2개의 ROM으로 구성되어 있다고 생각하면 된다. 또 CMOS로 만든 범용 프로세서는 같은 칩에 EEPROM이 만들어져 있는 것도 잊기 때문에, 이것을 사용하면, 3개의 칩으로 구성하여야 하지만, 한 개의 칩으로 구성할 수 있기 때문에 큰 장점이 된다.

● AVR 코어

AVR 마이컴·패밀리들은 일반적으로 플래시 메모리 크기에 따라 3부류로 나누어진다.

플래시 · 메모리의 크기가 1k~2k 바이트 범위인 것을 Tiny AVR

플래시 · 메모리의 크기가 2k~8k 바이트 범위인 것을 Classic AVR

플래시 · 메모리의 크기가 8k~128k 바이트 범위인 것을 Mega AVR

위의 3개의 패밀리가 전부 똑같은 구조로 되어 있기 때문에, 소프트웨어가 완전히 호환성이 있다. 예를 들어 Classic AVR로 설계하였는데, 프로그램 크기가 작게 되면, Tiny AVR로 쉽게 교체할 수 있다. 이런 유연성은 다른 마이컴에서는 없는 특징이다.

1. AT(MEGA)8535의 특징

1 고속 · 수행 능력과 저 · 전력 RISC 구조

- (1) 대부분 1클록 사이클에서 실행되는 118개의 강력한 명령어
- (2) 32×8 범용 워킹 레지스터
- (3) 8[MHz]에서 8[MIPS]까지 동작할 수 있다.

2 데이터와 비-휘발성(non-volatile)

- (1) ISP(In System Programmable)방법의 8k 바이트의 플래시 · 메모리
 - SPIシリ얼 인터페이스를 이용하여, ISP 방법으로 써넣을 수 있다.
 - 1000번까지 쓰기/지우기가 가능하다.
- (2) 100,000번까지 쓰기/지우기가 가능한 512바이트의 EEPROM
- (3) 소프트웨어 보호를 위해서 락(lock)기능이 있다.

3 주변 장치 특징

- (1) 8채널, 10비트 A/D 컨버터
- (2) 프로그래머블(Programmable) UART
- (3) 프리스케일러(Prescaler)와 카운터 모드가 있는 2개의 8비트 타이머/카운터
- (4) 프리스케일러, 비교기, 캡쳐(Capture)모드, 8/9/10 PWM 기능이 있는 16비트 타이머/카운터
- (5) 온 · 칩(On · chip) 발진기가 있는 프로그래머블 위치독 타이머
- (6) 온 · 칩 아날로그 비교기

4 특별한 기능

- (1) 파워 온(Powre on) 리셋 회로
- (2) 발진기와 카운터 모드가 있는 리얼 타임 클록(Real Time Clock)
- (3) 외부 혹은 내부 인터럽트 소스
- (4) 아이들(idle), 파워 절약(power save), 파워 다운(power down)모드의 상태로 될 수 있는 슬립 모드

5 CPU 속도 : 0~8[MHz]

2. AT(MEGA)8535의 내부 구조

AT(MEGA)8535는 32개의 범용 레지스터를 이용한 풍부한 명령을 갖고 있으며, 32개의 레지스터들은 ALU(Arithmetic Logic Unit)에 직접 연결되어 있어서, 1사이클내에서 한 명령으로 2개의 다른 레지스터를 액세스할 수 있다. 결과적으로 기존의 다른 CISC 마이크로 컨트롤러 보다 명령실행시간이 10배 정도는 빠른 효과를 얻게 된다.

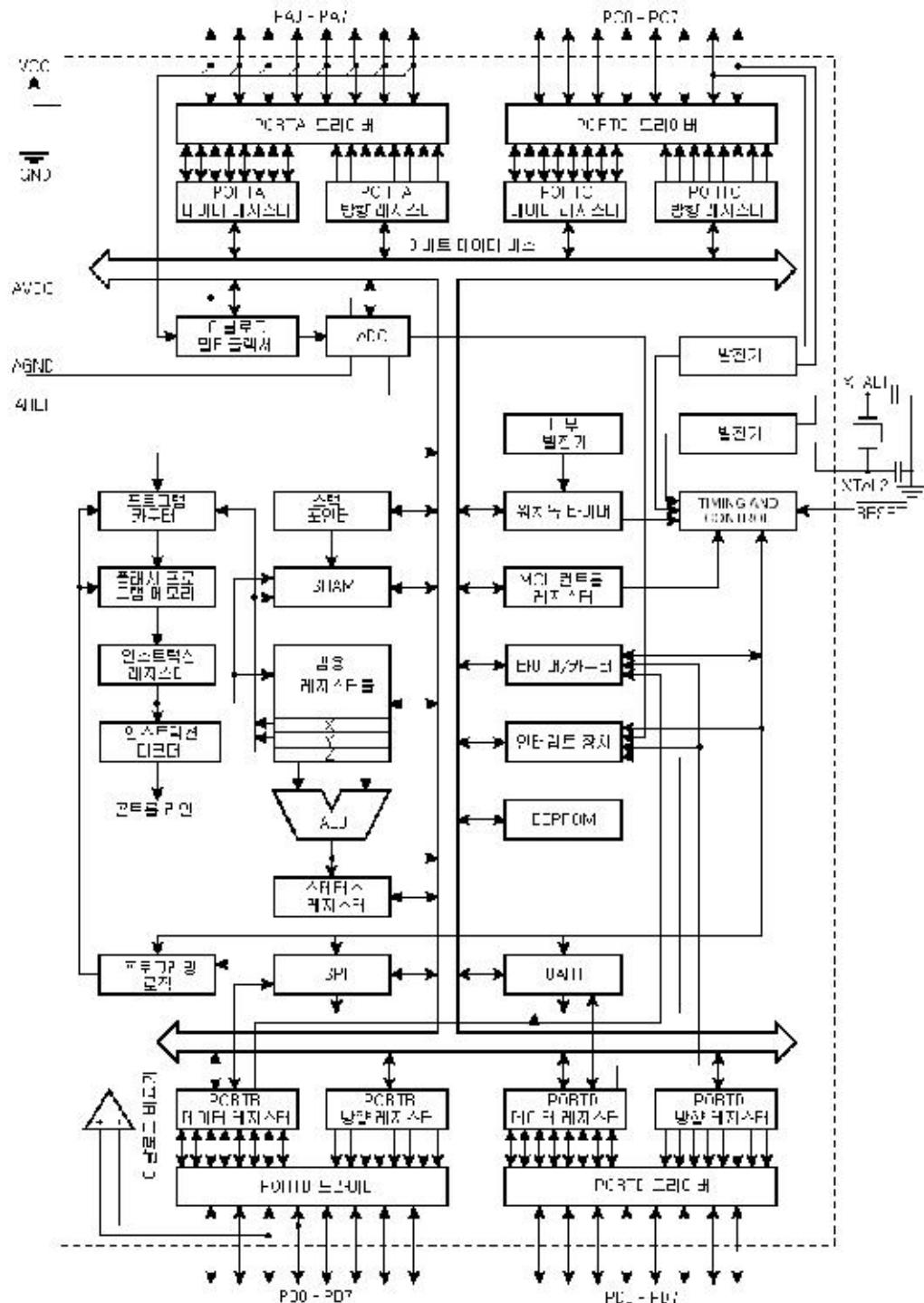
AVR 마이컴은 프로그램용 버스와 데이터 버스가 분리되어 있는 하버드(Harvard)구조로 구성되어 있으며, 프로그램 메모리는 2단계 파이프 라인(Pipeline)방법으로 실행된다. 즉 한 명령이 실행되는 동안, 다음 명령이 프로그램 메모리에서 미리 페치(Fetch)된다.

■ 범용 워킹 레지스터

범용 워킹 레지스터	
비트 번호→ 7	0 어드레스
R0	0x00
R1	0x01
R2	0x02
:	:
R13	0x0D
R14	0x0B
R15	0x0F
R16	0x10
R17	0x11
:	:
R26	0x1A X 레지스터 하위 바이트
R27	0x1B X 레지스터 상위 바이트
R28	0x1C Y 레지스터 하위 바이트
R29	0x1D Y 레지스터 상위 바이트
R30	0x1E Z 레지스터 하위 바이트
R31	0x1F Z 레지스터 상위 바이트

AT90S8535의 범용 워킹 레지스터

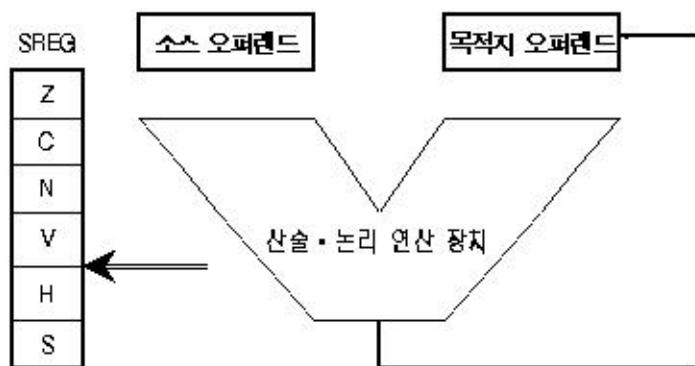
- 명령어에서 오퍼랜드로 사용하는 레지스터들에 모두 사용할 수 있으며, 모든 레지스터들은 1사이클로 액세스 할 수 있다.
- 각 레지스터들은 데이터 메모리 영역 0x00~0x1f로 매핑되어 있다.
- R26~R31 레지스터는 범용 레지스터 이외에 간접 번지 지정에서 어드레스 포인터로 사용되며 X, Y, Z 레지스터로 정의한다. 또 변위, 자동증가, 자동감소 기능이 있는 간접번지 지정에서도 사용된다.



AT90S8535의 내부 블록도

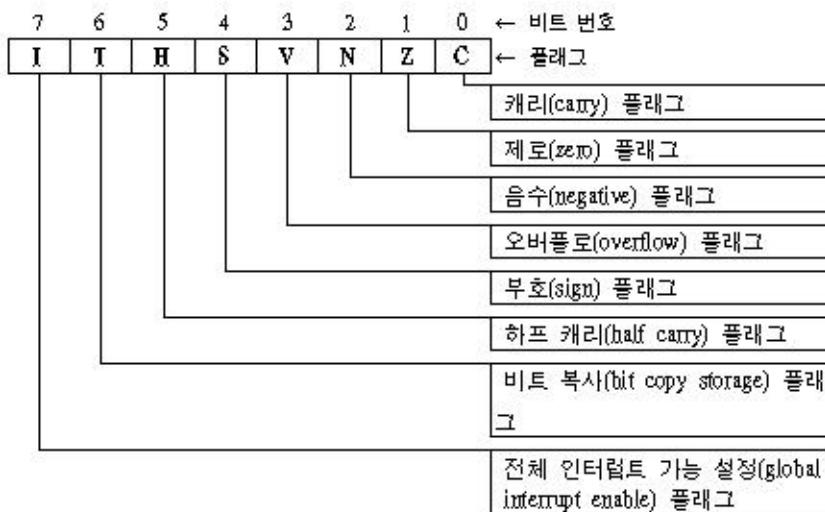
2 ALU(Arithmetic Logic Unit)

AVR 마이크로프로세서의 산술 연산 명령은 AVR 마이크로프로세서 내부에 있는 산술 · 논리 연산 장치(ALU : Arithmetic Logic Unit)에서 실행되며 아래 그림과 같이 소스 오퍼랜드와 목적지 오퍼랜드가 산술 연산 명령을 실행한 후 SREG(Status Register)의 각 플래그에 영향을 주며, 결과는 목적지 오퍼랜드에 저장된다.



산술·논리 연산 장치의 기본 동작

3 SREG(Status Register)

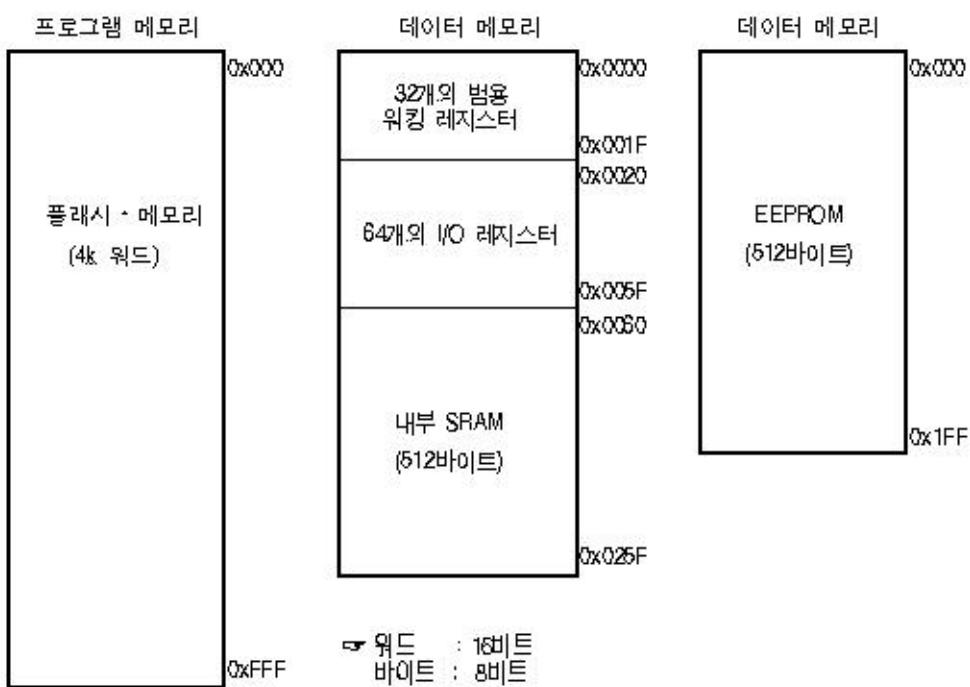


SREG(Status Register)

2. AT(MEGA)8535의 메모리 구조

AT(MEGA)8535의 메모리 구조는 아래 그림과 같으며, 각 메모리의 기능은 다음과 같다.

- 플래시 · 메모리 : 4k워드(16비트)의 크기이며, 사용자가 프로그램을 ISP방법으로 써넣는다.
- 32개의 범용 워킹 레지스터 : 어드레스 0x20 ~ 0x5f의 64개의 영역으로 되어 있으며, 컨트롤 레지스터, 타이머/카운터, A/D 컨버터, 다른 I/O 기능을 제어한다.
- 내부 SRAM : 512바이트 내부 SRAM은 데이터 저장 및 스택 등으로 사용한다.
- EEPROM : 512바이트로 구성된 EEPROM은 데이터 저장으로 사용한다.



AT90S8535의 메모리 맵

1. 플래시 · 메모리

AT(MEGA)8535는 프로그램 영역으로 플래시 · 메모리 8k바이트를 내장하고 있고, 모든 명령은 16비트 혹은 32비트로 되어 있기 때문에, 플래시 · 메모리의 크기는 $4k \times 16$ (4k워드 혹은 8k바이트)이 된다. 플래시 · 메모리는 100번 써넣기 · 지우기를 할 수 있다.

AT(MEGA)8535의 PC(Program Counter)는 12비트이므로, 프로그램 영역은 4096(4K워드)의 어드레스를 지정 할 수 있다.

2 SRAM 데이터 · 메모리

AT(MEGA)8535의 SRAM 메모리 구조는 608개의 데이터 메모리는 레지스터 파일, I/O 메모리, 내부 SRAM으로 되어 있으며, 처음 96바이트는 레지스터 파일 및 I/O 메모리, 다음 512바이트는 내부 SRAM으로 되어 있다.

레지스터 파일	데이터 메모리
R0	0x0000
R1	0x0001
R2	0x0002
:	:
R29	0x001D
R30	0x001E
R31	0x001F

I/O 레지스터	데이터 메모리
0x00	0x0020
0x01	0x0021
0x02	0x0022
:	:
0x3D	0x005D
0x3E	0x005E
0x3F	0x005F

내부 SRAM
0x0060
0x0061
:
0x025B
0x025F

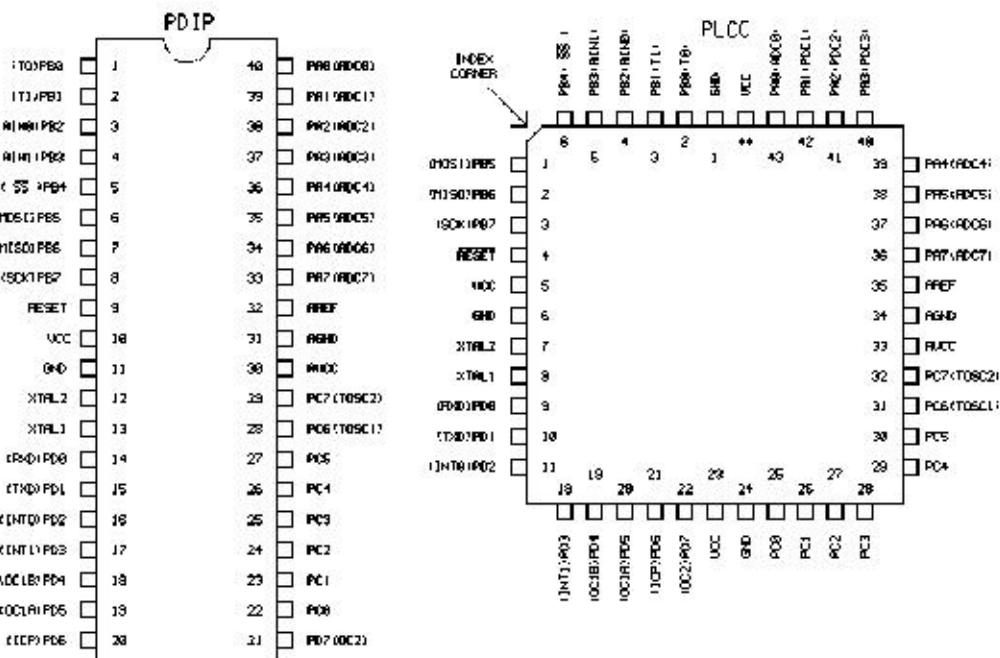
SRAM 구조

3 EEPROM 데이터 · 메모리

AT(MEGA)8535는 512바이트의 EEPROM으로 데이터를 읽기/쓰기를 할 수 있으며, 100,000번 읽기/쓰기가 가능하다. EEPROM으로 데이터를 읽기/쓰기하기 위해서는 EEPROM 어드레스 레지스터, EEPROM데이터 레지스터, EEPROM 컨트롤 레지스터를 이용하여야 한다.

3. AT(MEGA)8535의 핀의 기능

AT(MEGA)8535는 DIP(Dual In line Package), PLCC(Plastic Leaded Chip Carrier)의 형태가 있다.



AT(MEGA)8535의 핀 구조

1 포트 A : PA7~PA0/ADC7~ADC0

(1) 8비트 양·방향 I/O 포트

- 포트 핀은 내부적으로 풀업(pull-up) 저항이 연결되어 있다.
- 출력은 버퍼로 되어 있어서 20[mA]의 전류를 싱크(sink)할 수 있어, LED를 직접 구동할 수 있다.
- 입력으로 사용할 수 있다.

(2) A/D 컨버터의 채널 0~7입력으로도 사용한다.

(3) 리셋 조건에서는 클록이 동작하지 않더라도, 3-스테이트 상태가 된다.

2 포트 B : PB7~PB0

(1) 8비트 양·방향 I/O 포트

- 포트 핀은 내부적으로 풀업(pull-up) 저항이 연결되어 있다.
- 출력은 버퍼로 되어 있어서 20[mA]의 전류를 싱크(sink)할 수 있어, LED를 직접 구동할 수 있다.
- 입력으로 사용할 수 있다.

(2) 리셋 조건에서는 클록이 동작하지 않더라도, 3-스테이트 상태가 된다.

(3) 여러 종류의 다른 기능은 다음과 같다.

- PB0/T0 : 타이머 0의 입력
- PB1/T1 : 타이머 1의 입력
- PB2/AIN0 : 아날로그 비교의 플러스 입력
- PB3/AIN1 : 아날로그 비교의 마이너스 입력
- PB4/SS : SPI 슬레이브 선택 입력
- PB5/MOSI : SPI 기능에서 마스터 모드 때는 출력, 슬레이브 모드 때는 입력, 플래시 프로그램 메모리 읽기/쓰기에도 사용한다.
- PB6/MISO : SPI 기능에서 마스터 모드 때는 입력, 슬레이브 모드 때는 출력, 플래시 프로그램 메모리 읽기/쓰기에도 사용한다.
- PB7/SCK : SPI 기능에서 클록으로 사용된다. 플래시 프로그램 메모리 읽기/쓰기에도 사용한다.

3] 포트 C : PC7~PC0

(1) 8비트 양·방향 I/O 포트

- 포트 핀은 내부적으로 풀업(pull-up) 저항이 연결되어 있다.
- 출력은 버퍼로 되어 있어서 20[mA]의 전류를 싱크(sink)할 수 있어, LED를 직접 구동할 수 있다.
- 입력으로 사용할 수 있다.

(2) 리셋 조건에서는 클록이 동작하지 않더라도, 3-스테이트 상태가 된다.

(3) PO6/TOSC1, PC7/TOSC2는 타이머/카운터 2에서 32.768[Hz] 발진기를 연결하는데 사용된다.

4] 포트 4 : PD7~PD0

(1) 8비트 양·방향 I/O 포트

- 포트 핀은 내부적으로 풀업(pull-up) 저항이 연결되어 있다.
- 출력은 버퍼로 되어 있어서 20[mA]의 전류를 싱크(sink)할 수 있어, LED를 직접 구동할 수 있다.
- 입력으로 사용할 수 있다.

(2) 리셋 조건에서는 클록이 동작하지 않더라도, 3-스테이트 상태가 된다.

(3) 여러 종류의 다른 기능은 다음과 같다.

- PD0/RXD : UART의 시리얼(serial) 입력
- PD1/TXD : UART의 시리얼 출력
- PD2/INT0 : 외부 인터럽트 입력 INT0으로 사용
- PD3/INT1 : 외부 인터럽트 입력 INT1로 사용
- PD4/OC1B : 타이머 1의 비교 B 출력
- PD5/OC1A : 타이머 1의 비교 A 출력
- PD6/ICP : 타이머 1의 입력
- PD7/OC2 : 타이머 2의 비교 출력

5 **RESET** : 입력, 액티브 로우(low)

이 편에 “L”이 되면, AT(MEGA)8535는 리셋되며, 리셋 신호는 50[ns]보다 길어야 한다. 플래시 메모리로 읽기/쓰기 할 때는 “L”을 유지하여야 한다.

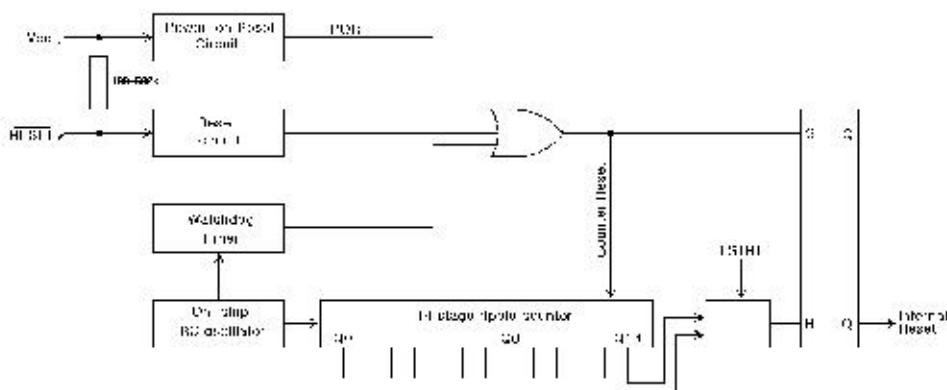
(1) AT(MEGA)8535는 다음과 같이 3개의 리셋 소스를 갖고 있다.

- 파워 · 온 리셋(power-on reset) : 공급 전압이 파워 · 온 리셋의 스레스홀드(threshold) 전압 이하가 된다.
- 외부 리셋(external reset) : **RESET** 편이 50[ns] 이상 “L”레벨이 되면 리셋된다.
- 워치독 리셋(watchdog reset) : 워치독 타이머 주기가 오버되면 리셋되고, 워치독도 리셋된다.

(2) AT(MEGA)8535의 리셋 상태는 다음과 같다.

- 모든 I/O 레지스터는 초기 설정 값으로 된다.
- 프로그램은 0x0000 번지부터 실행된다.

(3) 아래 그림은 AT(MEGA)8535 내부에 내장된 리셋 회로이다.



내부 리셋 회로

6 **XTAL1, XTAL2** : 클록 입 · 출력

7 **VCC, GND** : 디지털 전원

8 AVCC

포트 A와 A/D 컨버터의 전원 입력 단자, A/D 컨버터를 사용하지 않을 경우에는 V_{OCC}에 연결하고, A/D 컨버터를 사용할 경우는 필터를 거쳐서 V_{OCC}에 연결한다.

9 AREF

A/D 컨버터의 아날로그 기준 입력, A/D 컨버터의 동작을 위해서 AV_{OCC}와 AGND 사이의 전압을 연결한다.

10 AGND

A/D 컨버터에서 사용하는 접지, 보통 GND에 연결한다.

제 3장 PIC : PIC16F874/7

PIC(Peripheral Interface Controller)은 크게 3가지로 분류한다.

- (1) 12비트 구조(low range)
- (2) 14비트 구조(middle range)
- (3) 16비트 구조(high range)

1 12 비트 구조의 PIC

12비트 구조(low range)의 PIC은 PIC12C508(8핀), PIC16C54, PIC16C57(28핀)등이 많이 사용되었으며, ROM의 데이터 폭은 12비트로 되어 있어서, 프로그램을 만들 때 제약이 많아서 사용하기에 불편하지만 가격이 저렴하다.

2 14 비트 구조의 PIC

14비트 구조(middle range)의 PIC은 현재 가장 많이 사용되고 있는 PIC16F84가 있으며, PIC 시리즈의 중심인 제품이다.

3 16 비트 구조의 PIC

16비트 구조(high range)의 PIC은 PIC18C 시리즈이며 14비트 구조의 PIC보다 동작 속도, 프로그램 메모리, 데이터 메모리, 명령어, 하드웨어 부분 등을 많이 보강하여 높았다.

1. PIC16F87x의 특징

■ 마이크로 컨트롤러 코어(Micro-controller core)의 특징

- 고 성능의 RISC CPU
- 35개의 명령
- 점프(2사이클)명령을 제외하고, 모든 명령은 1사이클로 실행
- 동작 속도 : DC~20[MHz]의 클록 입력(DC~20[ns] 명령 사이클)
- 8k × 14 워드 플랫 메모리, 368 × 8바이트 데이터 메모리, 256 × 8바이트 데이터 메모리
- PIC16C73B/74B/76/77 편의 호환성
- 인터럽트 14개
- 8개의 하드웨어 스택
- 직접, 간접, 상대 번지 지정
- 파워 온 리셋(POR : Power-On Reset)
- 파워 업 타이머(PWRT : Power-up Timer)와 오실레이터 스타트 업 타이머 (OST : Oscillator Start-up Timer)
- 워치독 타이머(WDT : Watchdog Timer)
- 프로그램 보호
- 파워 절약을 위한 슬립 모드
- 발진기 오션 선택 가능
- 저 전력, 고속 CMOS FLASH/EEPROM
- 2핀을 이용한 인 서킷 시리얼 프로그래밍 (ICSP : In-Circuit Serial Programming)
- 5[V] 전원으로 프로그래밍
- 2핀을 이용한 인 서킷 디버깅
- 프로세서로 프로그램 메모리 리드/라이트 가능
- 20[V]~5.5[V] 동작 가능
- 높은 싱크/소스 전류 : 25[mA]

표 1-1. PIC16F87x의 사양

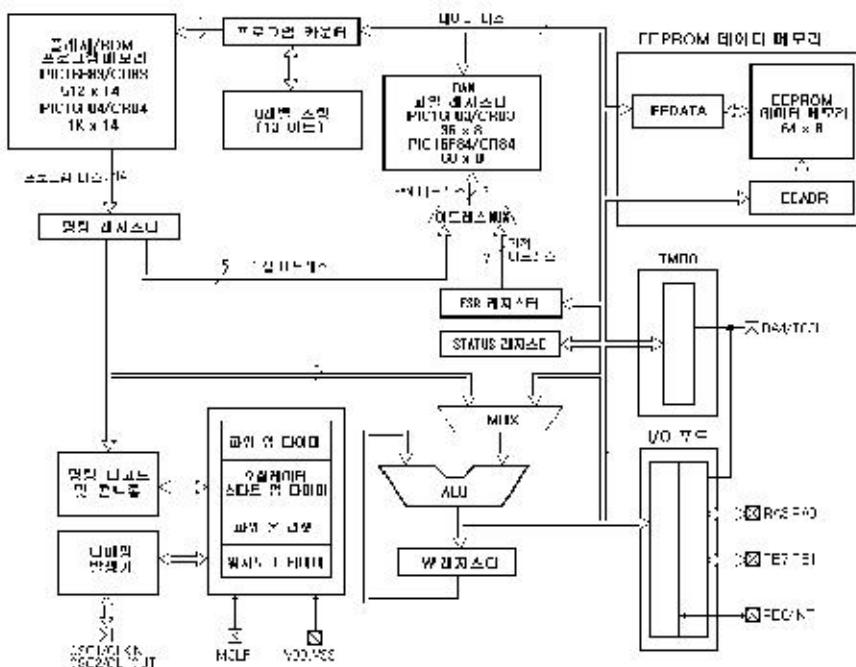
항목	PIC16F873	PIC16F874	PIC16F876	PIC16F877
동작 주파수	DC~20[MHz]	DC~20[MHz]	DC~20[MHz]	DC~20[MHz]
리셋(자연 회로)	POR, BOR, MCLR, WDT(PWRT,OST)	POR, BOR, MCLR, WDT(PWRT,OST)	POR, BOR, MCLR, WDT(PWRT,OST)	POR, BOR, MCLR, WDT(PWRT,OST)
플래시 메모리	4K×14비트	4K×14비트	8K×14비트	8K×14비트
데이터 메모리	192바이트	192바이트	368바이트	368바이트
EPPROM 데이터 메모리	128 바이트	128 바이트	256 바이트	256 바이트
인터럽트	13	14	13	14
I/O 포트	포트 A, B, C	포트 A, B, C, D, E	포트 A, B, C	포트 A, B, C, D, E
타이머	3	3	3	3
캡처/컴퍼어/PWM	2	2	2	2
직렬 인터페이스	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
병렬 인터페이스	—	—	—	PSP
10비트 A-D컨버터	5채널	5채널	8채널	8채널
밀링 수	35	35	35	35

2 주변 장치(Peripheral)의 특징

- 타이머 0 : 8비트 프리스케일러(prescaler)가 있는 8비트 타이머/카운터
- 타이머 1 : 8비트 프리스케일러가 있는 16비트 타이머/카운터, 슬립 모드 동안 외부 클록으로 동작할 수 있다.
- 타이머 2 : 8비트 주기 레지스터, 프리스케일러, 포스트스케일러(postscaler)가 있는 8비트 타이머/카운터
- 2개의 캡쳐(Capture), 비교(Compare), PWM 모듈
 - 캡쳐는 16비트, 최대 분해능(resolution)은 125[ns]
 - 비교는 16비트, 최대 분해능은 200[ns]
 - PWM의 최대 분해능은 10비트
- 10비트 여러 채널 A/D(Analog-to-Digital)컨버터
- 동기식シリ얼 포트(SSP : Synchronous Serial Port)
- 비동기식シリ얼 포트
- 8비트, RD, WR, CS(40핀/44핀 만 가능)가 있는 병렬 슬레이브 포트(PSP : Parallel Slave Port)
- 브라운 아웃 리셋(BOR : Brown-out Reset)에 사용되는 브라운 아웃 검출 회로

2. PIC16F87x의 내부 구조

PIC16F874, PIC16F877에 전체 내부 블록도를 나타내었다.



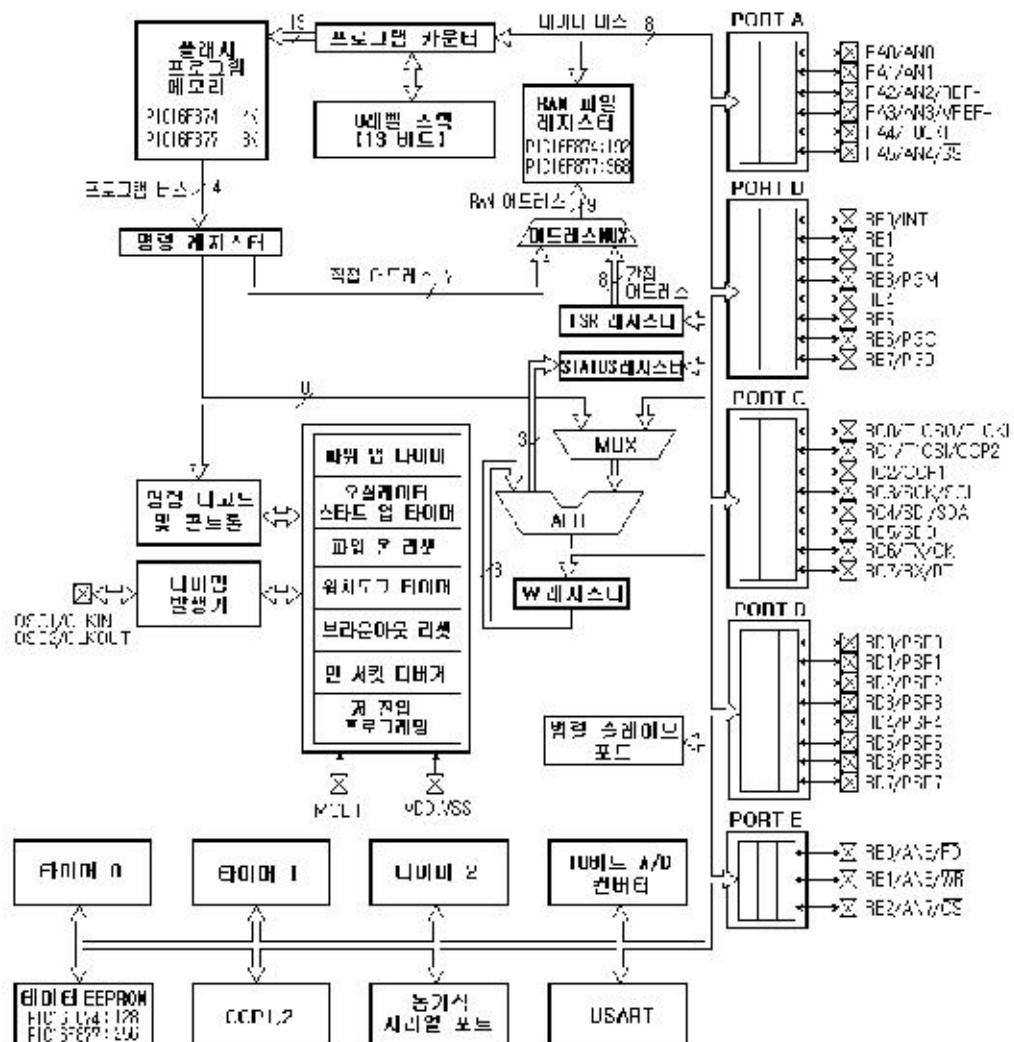
PIC16F87x의 내부 구조

1 ALU, W 레지스터, 스테터스 레지스터

ALU(Arithmetic Logic Unit : 산술 논리 연산 장치)는 8비트 산술 논리 연산 장치이며, W 레지스터에 있는 데이터와 레지스터 파일의 내용을 산술 혹은 논리 연산을 한다.

W(Working) 레지스터는 보통 다른 CPU에서 어큐му레이터(accumulator)에 해당되며, 데이터 전송과 ALU의 연산에 사용되는 임시 저장용 레지스터이다.

STATUS 레지스터는 산술 연산의 상태를 나타내는 플래그(flag)들로 구성되어 있다.



3 FSR 레지스터

CPU 기능을 제어하는 각종 레지스터 이 외에, 사용자가 자유롭게 사용할 수 있는 범용 레지스터이다.

4 프로그램 카운터

프로그램 카운터(PC : Program Counter)는 13비트 길이의 레지스터로, 하위 8비트는 레지스터 파일에 있는 PCL 레지스터로 읽기/쓰기가 가능하며, 상위 5비트는 직접 읽기/쓰기를 할 수 없다. PC의 상위 5비트는 레지스터 파일에 있는 PCLATCH 레지스터를 이용하여 변경할 수 있지만, 실제로 PC에 써넣는 CALL 명령, GOTO 명령 또는 PCL 레지스터로 써넣기를 할 때만 써넣을 수 있다. PC는 리셋되면 0000H로 된다.

5 스택

보통 CPU에서는 PC를 CALL 명령과 인터럽트가 발생하면 스택 포인터가 지시하는 메모리 공간으로 대피시키지만, PIC16F84에서는 블록도에서 보면 알 수 있듯이 8개 스택 영역으로 대피한다. 따라서 레지스터 파일의 범용 레지스터(RAM)등에 스택 영역을 확보 할 필요는 없다. 그러므로 스택 영역에 저장되어 있는 PC 값을 읽기/쓰기를 할 수 없다.

6 프로그램 메모리

프로그램을 저장하는 메모리로, 프로그램 카운터로 액세스 할 수 있는 8k × 14 프로그램 메모리가 있다. PIC 시리즈에서 PIC에 내장되어 있는 프로그램 메모리를 구별하는 방법은 아래와 같이 C, CR, F로 구별한다.

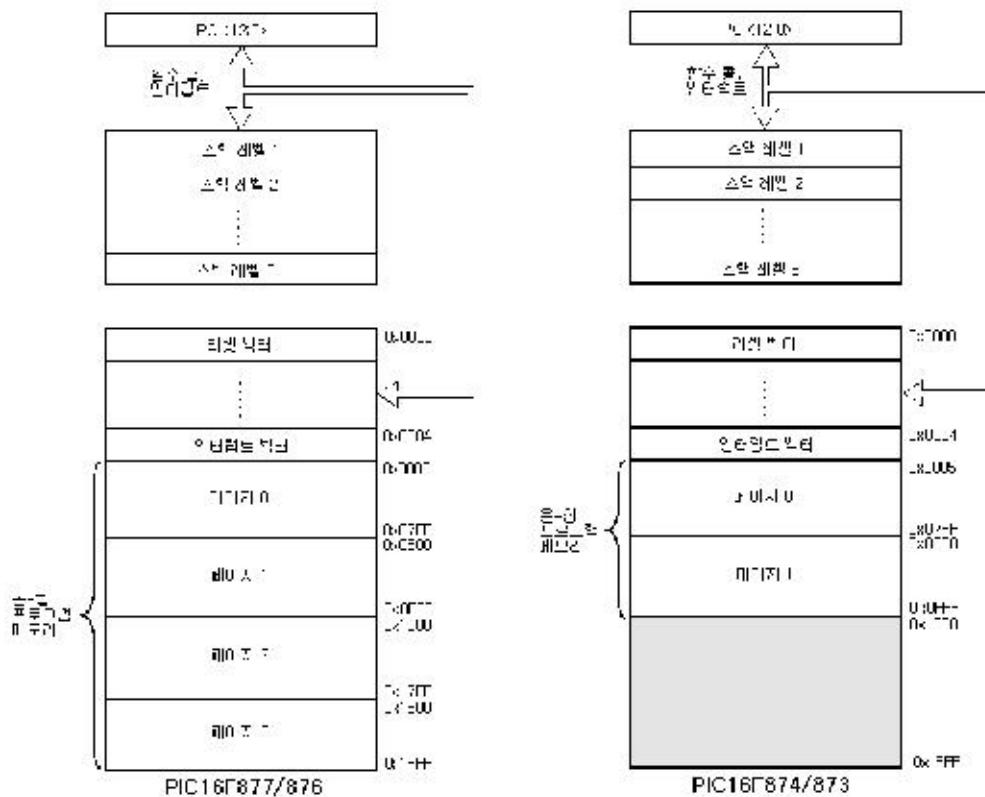
- PIC16Cxxx : C는 CMOS형 자외선 소거형 EEPROM
- PIC16CRxxx : CR은 CMOS형 마스크 ROM
- PIC16Fxxx : F는 플래시 메모리(Flash memory)

7 IR 레지스터

IR(Instruction Register)레지스터는 명령의 OP(Operation code)가 저장되며, 이 IR 레지스터의 내용이 “Instruction Decode & Control”에서 해독되고, 그 명령에 맞는 제어를 하게 된다.

3. PIC16F87x의 메모리 구조

PIC16F87x의 메모리는 블록도에서 보면 알 수 있듯이 프로그램 메모리와 데이터 메모리(범용 RAM, SFR, 데이터 EEPROM)가 있으며, 각 메모리는 각자 자신의 버스를 갖고 있어서 같은 클록 사이클에서 동시에 읽기/쓰기를 할 수 있다.



PIC16F87x의 프로그램 메모리 블록도

1 프로그램 메모리

프로그램을 저장하는 메모리로, 프로그램 카운터로 액세스 할 수 있는 $8k \times 14$ 프로그램 메모리가 있다.

2 데이터 메모리

표 1-2. 뱅크 선택

RP1	RP0	뱅크	설명
0	0	0	데이터 메모리는 범용 레지스터와 SFR로 구성된 여러 뱅크로 나누어지며 RP1, RP0으로 각각 선택된다. 각 뱅크는 128(0x7F) 바이트 크기로 되어 있고, 각 뱅크의 하위는 SFR(Special Function Register)로 구성되고, SFR의 위는 범용 레지스터로 사용되는 레지스터 파일이라 부른다.
0	1	1	
1	0	2	
1	1	3	

(1) 범용 레지스터

범용 레지스터(GPR : General Purpose Register)는 SFR 다음 영역으로 할당되어 있으며, 레지스터 파일 혹은 GPR 이라 부르며, 직접전지 지정 혹은 PSR(File Select Register)레지스터를 이용하여 간접번지 지정으로 액세스 할 수 있어서, 현재 뱅크에 관계없이 같은 위치에 읽기/쓰기를 할 수도 있다.

• PIC16P873/874

각 뱅크에 96 바이트의 용량을 갖고 있으며, 뱅크 3은 뱅크 0, 뱅크4는 뱅크 1에 미러(mirror)되어 있기 때문에 실제로 사용할 수 있는 용량은 $96 \times 2 = 192$ 바이트가 된다.

• PIC16P876/877

뱅크 0에 96 바이트, 뱅크 1의 80바이트, 뱅크2, 3의 96바이트의 총 368 바이트로 구성되어 있으며, 뱅크 1, 뱅크 2, 뱅크3의 상위 16 바이트는 뱅크 1에 미러(mirror)되어 있어서 뱅크 0의 0x70~0x7F의 내용과 같다.

■ 구현되지 않은 메소드 제거가 되면 '이' 끝으면 '고' (1) - 케이스는 그대로继续保持되는 구현되지 않는다.
(2) - 드리워진 경우스티가 된다.

PIC16F876/7의 레지스터 파일 맵

2부/제 3장 PIC ; PIC16F874/7

■ 구현되지 않은 메이플 게임의 위치 : 등으면 '0' (1) 드래프트 게임 (2) 드래프트 게임에서만 구현되지 않는다.
(3) 드라마틱한 챔피언 스타터가 대다수 (4) 게임 솔루션은 제작된 것

PIC16F873/4 레지스터 앱

(2) SPR

SPR(Special Function Registers) 레지스터들은 CPU와 주변 장치들을 사용자가 원하는 기능으로 동작하도록 제어하는데 사용한다. 이 레지스터들은 S(Static)RAM으로 구성되어 있으며, "core"기능과 관계있는 것들과 주변 장치의 동작에 관계된 것으로 분류한다. SPR 레지스터들도 뱅크로 되어 있으므로, STATUS 레지스터의 RP1, RP0을 이용하여 원하는 SPR 레지스터를 선택하여야 한다.

3 데이터 메모리 EEPROM

8비트 길이의 데이터 메모리가 있으며, 이 데이터 메모리는 블록도에서도 이해할 수 있지 만, 프로그램 메모리 및 레지스터 파일과는 별도로 되어 있다. 데이터 메모리 EEPROM은 프로그램 메모리와 똑같이 써넣기/읽기를 하지만, 프로그램으로도 써넣기/읽기를 할 수 있다.

3. PIC16F87x의 펈의 기능**1 포트 A : RA0~RA7, 양방향 I/O포트, TTL 레벨**

- RA0/AN0 : RA0핀은 아날로그 입력 0으로 사용한다.
- RA1/AN1 : RA1핀은 아날로그 입력 1로 사용한다.
- RA2/AN2/VREF- : RA2핀은 아날로그 입력 2로 사용하거나, 마이너스 아날로그 기준 전압으로 사용한다.
- RA3/AN3/VREF+ : RA3핀은 아날로그 입력 3으로 사용하거나, 플러스 아날로그 기준 전압으로 사용한다.
- RA4/T0CKI : RA4 핀을 타이머 0의 클록 입력으로 사용하며, 출력으로 사용할 때는 오픈 드레인(open drain) 형태로 되어 있다.
- RA5/SS/AN4 : RA5핀은 아날로그 입력 혹은 동기식 시리얼 포트에서 슬레이브 선택에 사용한다.

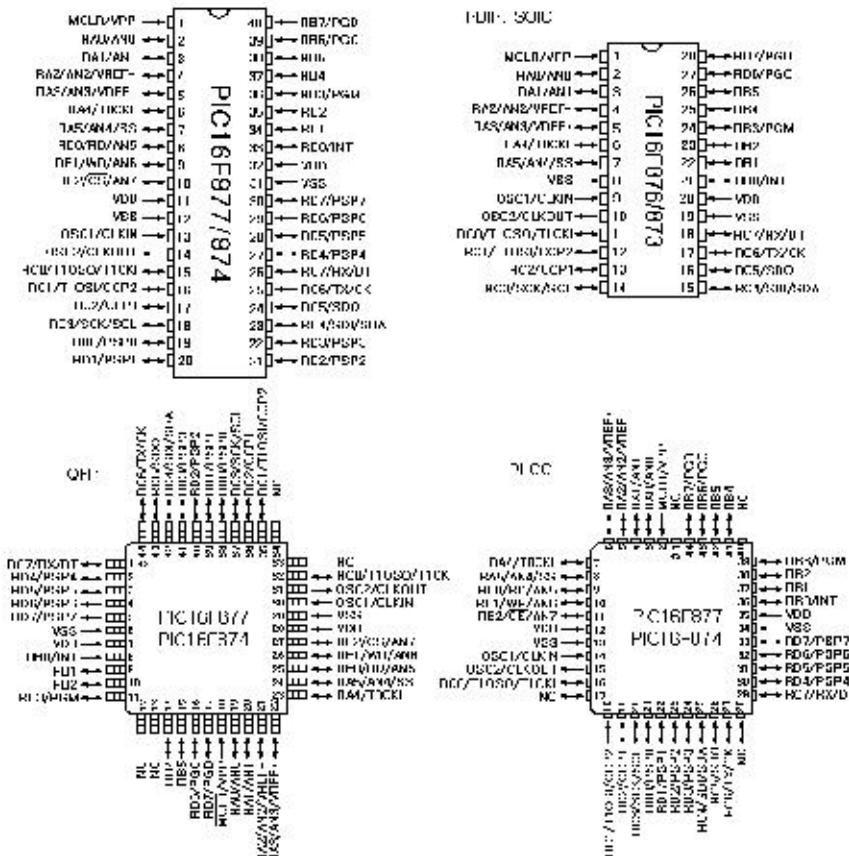


그림 1-11. PIC16F87x의 핀 배치도

2 포트 B : RB0~RB7, 양방향 I/O포트, TTL 레벨

포트 B는 모두 입력으로 사용할 경우, 내부적으로 약한 풀업이 될 수 있도록 소프트웨어로 선택이 가능하다.

- RB0/INT : 외부 인터럽트 입력 편으로 사용할 수 있다.
- RB1 : I/O 포트
- RB2 : I/O 포트
- RB3/PGM : RB3은 낮은 전압 프로그래밍 입력으로 사용할 수 있다.
- RB4 : 편의 입력 상태가 변하면 인터럽트를 요청할 수 있다.

- RB5 : 핀의 입력 상태가 변하면 인터럽트를 요청할 수 있다.
- RB6/PGC : 핀의 입력 상태가 변하면 인터럽트를 요청 할 수 있으며, 인 서킷 디버거(In Circuit Debugger)로 사용한다. PIC16F87x를 직렬(serial)로 써넣을 경우, 직렬 클록을 입력(시미트 트리거)하는 핀으로 사용한다.
- RB7/PGD : 핀의 입력 상태가 변하면 인터럽트를 요청 할 수 있으며, 인 서킷 디버거(In Circuit Debugger)로 사용한다. PIC16F87x를 직렬(serial)로 써넣을 경우, 직렬 클록을 입력(시미트 트리거)/출력 핀으로 사용한다.

■ 3 포트 C : RC0~RC7, 포트 C의 양방향 I/O포트, 시미트 트리거 입력

- RC0/T1OSO/T1CKI : RC0는 타이머 1 발진기 출력 혹은 타이머 1 클록 입력으로 사용한다.
- RC1/T1OSI/CCP2 : RC1은 타이머1 발진기 입력 혹은 캡쳐 2 입력/컴페어 2 출력/PWM2 출력으로 사용한다.
- RC2/CCP1 : RC2는 캡쳐 1 입력/컴페어 1 출력/PWM 1 출력으로 사용된다.
- RC3/SCK/SCL : RC3은 동기식 직렬 클록 입력/SPI와 I²C모드에서 출력으로 사용한다.
- RC4/SDI/SDA : RC4는 SPI 데이터 입력(SPI 모드) 혹은 데이터 I/O(I²C모드)로 사용한다.
- RC5/SDO : RC5는 SPI 데이터 출력(SPI 모드)으로 사용한다.
- RC6/TX/CK : RC6은 비동기식 전송 혹은 동기식 클록으로 사용한다.
- RC7/RX/DT : RC7은 비동기식 수신 혹은 동기식 데이터로 사용한다.

■ 4 포트 D : RD0/PSP0~RD7/PSP7, 포트 C의 양방향 I/O포트

다른 CPU에서 PIC16F874/877을 I/O 포트로 사용하기 위한 병렬 슬레이브(slave)포트, 범용 I/O 포트로 사용할 때에는, 시미트 트리거 입력, 병렬 슬레이브 모드로 사용 할 때에는 TTL 출력이 된다.

4 포트 E : RE0~RE2, 포트 E의 양방향 I/O포트

범용 I/O 포트로 사용할 때에는 시미트 트리거 입력, 병렬 슬레이브 모드로 사용할 때에는 TTL 출력이 된다.

- RE0/ \overline{RD} /AN5 : RE0은 병렬 슬레이브 모드에서 읽기 제어신호, 아날로그 입력 5로 사용한다.
- RE1/ \overline{WR} /AN6 : RE1은 병렬 슬레이브 모드에서 쓰기 제어신호, 아날로그 입력 6으로 사용한다.
- RE2/ \overline{CS} /AN6 : RE2은 병렬 슬레이브 모드에서 칩 선택 제어 신호, 아날로그 입력 7로 사용한다.



3부 MDA-Multi MICOM USB 예제

MDA-Multi MICOM USB보드에 AT89S51, AT(MEGA)8535, PIC16F87x의 각 CPU 보드를 이용하여 각각의 예제를 만들었습니다.

- 제 1장 I/O 포트 연습
- 제 2장 인터럽트 및 EEPROM
- 제 3장 타이머 및 캐릭터 LCD
- 제 4장 데이터 통신
- 제 5장 D/A, A/D 컨버터
- 제 6장 시리얼 통신
- 제 7장 도트 매트릭스
- 제 8장 모터 제어
- 제 9장 통신
- 제 10장 센서
- 제 11장 그래픽 LCD

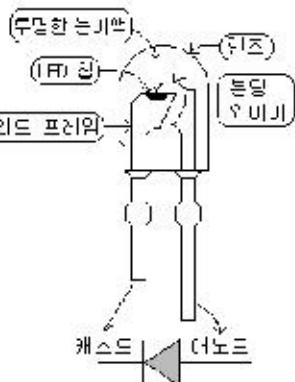
* 2장 ~ 11장 예제 파일은 MDA-MULTI MICOM USB CD-ROM의 Example을 참조하시기 바랍니다.

제 1장. I/O 포트 연습

1. LED 점등하기

- LED의 구조

LED는 광을 발산하는 본체는 LED 칩이다. LED 칩과 애노드(anode) · 리드 사이는 Ø25~30μm 의 금속선이 접속되어, 그곳에서 광을 적절히 출력하기 위해서 LED 칩은 투명한 송진동으로 렌즈의 가운데를 매워 놓았다. 렌즈의 형상과 송진의 종류는 아래와 같고 또 특성이 다른 LED를 만들기도 한다.

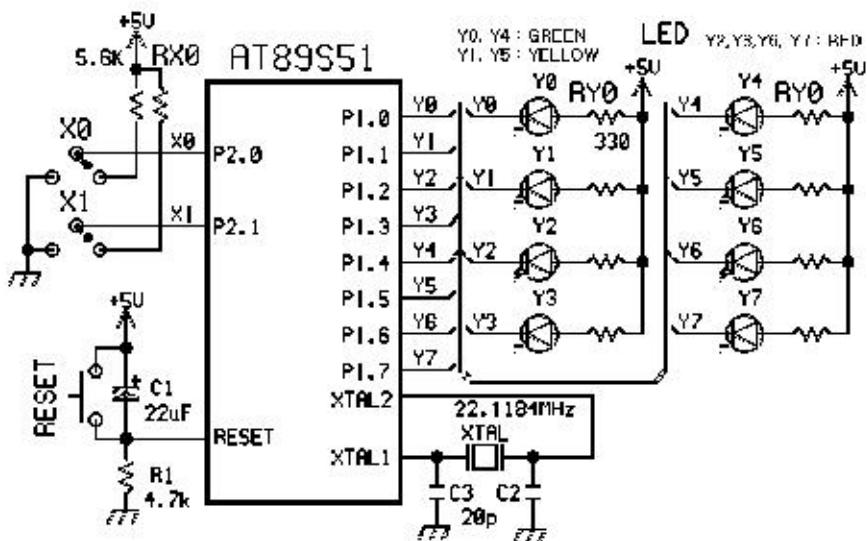


- (1) 무색 투명 렌즈
- (2) 발광색과 같은 송진으로 착색하고, 점등하지 않을 경우에 발광색을 판별할 수 있다.
- (3) 송진에 광 산란제를 넣어서 발광 면적을 증가시켜서 부드럽게 발광을 하도록 한다.

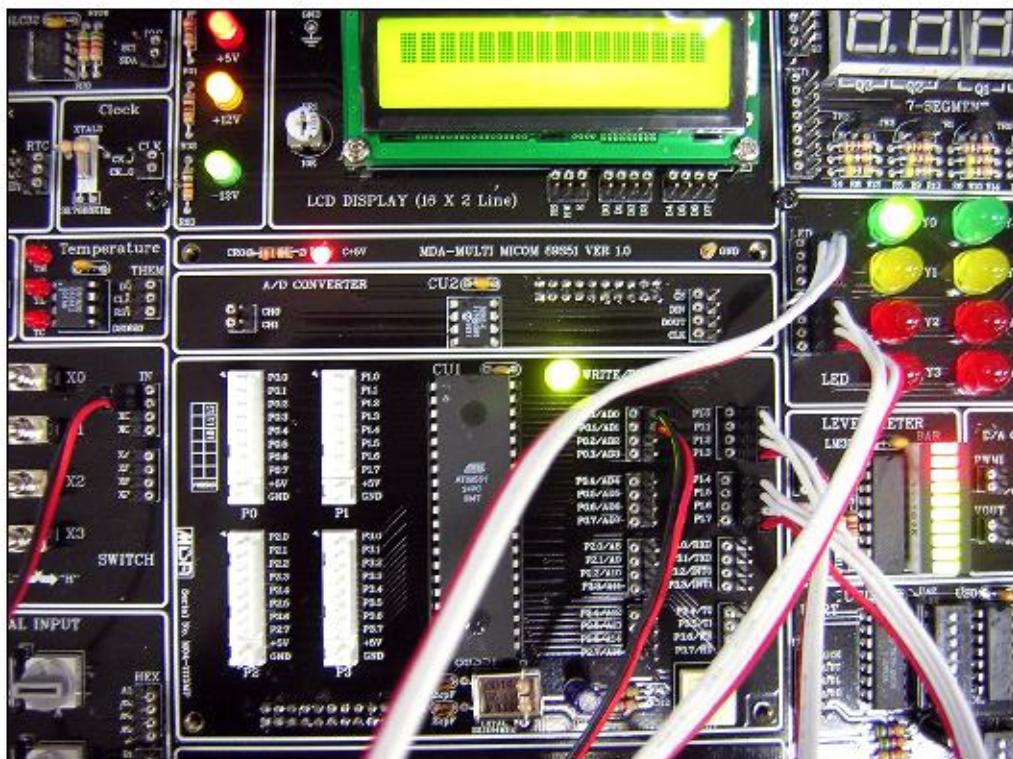
AT89S51

아래의 회로를 이용하여 LED를 X0, X1의 상태에 따라 동작시킨다.

X1	X0	LED Y0~Y7의 상태
0	0	LED 모두 OFF
0	1	LED Y0 → Y7의 순서대로 ON
1	0	LED Y0 ← Y7의 순서대로 ON
1	1	LED Y0 → Y7의 순서대로 ON



LED 점등 연습에 사용할 회로



LED 점등 연습을 위해 연결된 회로

● 프로그램 : LED1.C

```
*****
* MDA-MULTI MICOM
* MCU : MCS51
* COMPILER : IAR
* FILE NAME : LED1.C
*****
```

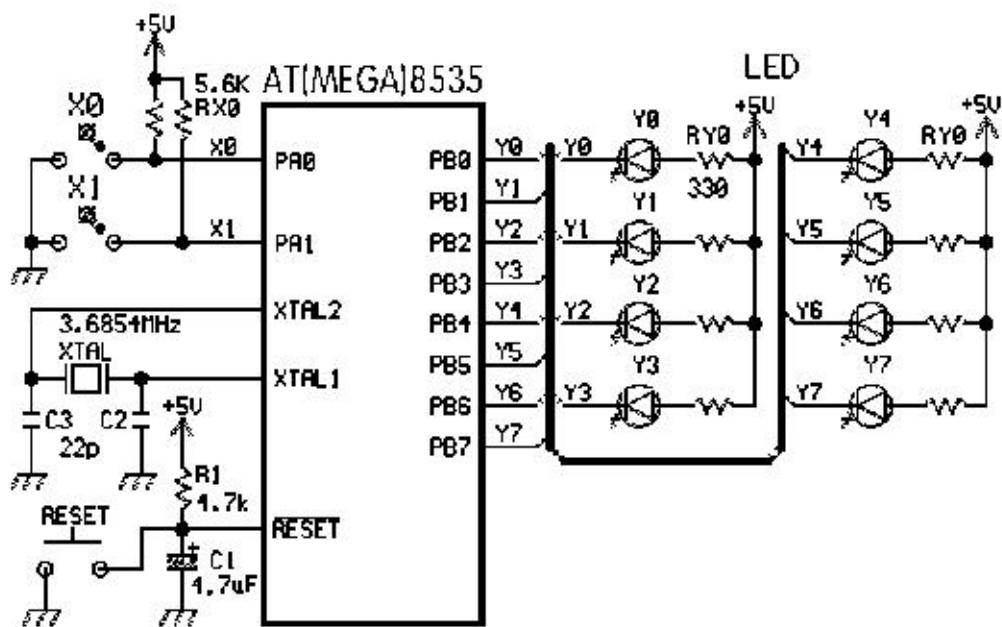
```
#include <i89s51.h>
unsigned char temp; // LED ON/OFF 저장 전역 변수
// 일정 시간 지연 함수
void delay (long i)
{
    while (i--);
}
// 시프트 레프트
void SHIFTL (void)
{
    if(temp == 0xff) { // 이 동작이 처음인지 검사
        temp = 0xfe; // 초기 값 저장
    }
    else {
        temp = (temp << 1)|0x01; // LED ON 값 업데이트
        if (temp == 0xff) temp = 0xfe; // 마지막 LED가 ON인 경우 초기 값 저장
    }
    delay (60000); //일정 시간 지연
}
// 시프트 라이트
void SHIFTR (void)
{
    if(temp == 0xff) { // 이 동작이 처음인지 검사
        temp = 0x7f; // 초기 값 저장
    }
    else{
        temp = (temp >> 1)|0x80; // LED ON 값 업데이트
        if (temp == 0xff) temp = 0x7f; // 마지막 LED가 ON인 경우 초기 값 저장
    }
}
```

```
    delay (60000); //일정 시간 지연
}
// 메인
void main(void)
{
    temp = 0x1f; // LED ON/OFF 초기 값
    P2 = 0xff;
    do{
        P1 = temp; // P1으로 출력
        if( P2_0 ) SHIPTL(); // X0 스위치="1"인 경우
        else if( P2_1 ) SHIPTR(); // X1 스위치="1"인 경우
        else temp = 0xff; // X0,X1 스위치="1"인 경우
    }while(1); // 무한 루프
}
```

2 AVR

아래의 회로를 이용하여 LED를 X0, X1의 상태에 따라 동작시킨다.

X1	X0	LED Y0~Y7의 상태
0	0	LED 모두 OFF
0	1	LED Y0 → Y7의 순서대로 ON
1	0	LED Y0 ← Y7의 순서대로 ON
1	1	LED Y0 → Y7의 순서대로 ON



LBD 점등 연습에 사용할 회로

- 프로그램 : LED1.C

```
*****  
* MDA-MULTI MICOM AVR  
* MCU : AVR  
* COMPILER : IAR  
* FILE NAME : LED1.C  
*****  
  
#include "io8635.h"  
#define X0 PINA_Bit0 // X0핀 정의  
#define X1 PINA_Bit1 // X1핀 정의  
unsigned char temp; // LED ON/OFF 저장 전역 변수  
// 일정 시간 지연 함수  
void delay (unsigned int i)  
{  
    while (i--);  
}  
// 시프트 레프트  
void SHIFTL (void)  
{  
    if(temp == 0xff) // 처음인 경우 검사  
    { temp = 0xfe; // 초기 값 저장  
        return; } // 리턴  
    else {  
        temp = (temp << 1)|0x01; // LED ON 값 업데이트  
        if (temp == 0xff) temp = 0xfe; // 마지막 LED가 ON인 경우 초기 값 저장  
    }  
}  
// 시프트 라이트  
void SHIFTR (void)  
{  
    if (temp == 0xff) // 처음인 경우 검사  
    { temp = 0x7f; // 초기 값 저장  
        return; } // 리턴  
    else{  
        temp = (temp >> 1)|0x80; // LED ON 값 업데이트
```

```

    if (temp == 0xff) temp = 0x7f; // 마지막 LED가 ON인 경우 초기 값 저장
}
}

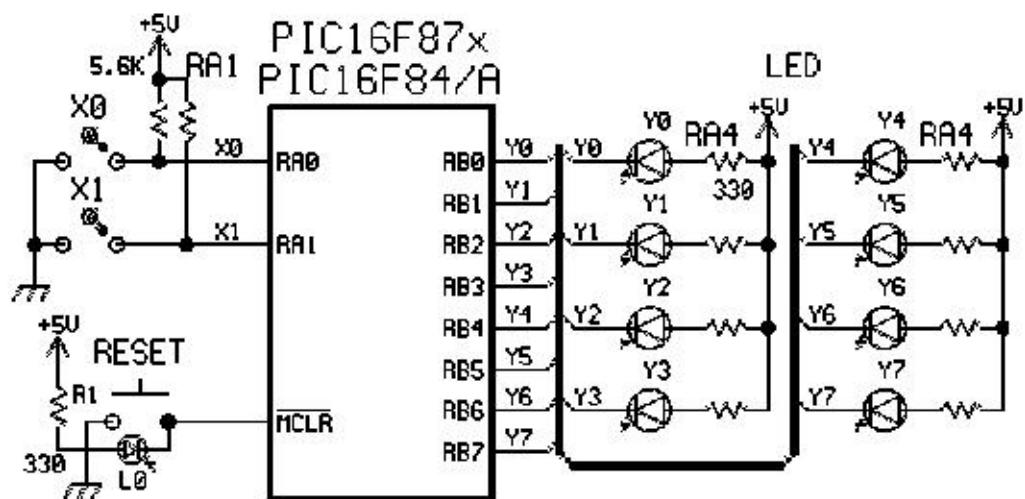
// 메인
void main(void)
{
    DDRB = 0xff; // B 포트 출력
    DDRA = 0x00; // A 포트 입력
    temp = 0xff; // LED ON/OFF 초기 값
    do{
        PORTB = temp; // PORTB로 출력
        delay (65000); // 일정 시간 지연
        if(X0) SHIFTL(); // X0 스위치="1"인 경우
        else if(X1) SHIFTRO(); // X1 스위치="1"인 경우
        else temp = 0xff; // X0, X1 스위치="0"인 경우
    }while(1); // 무한 루프
}

```

3 PIC

아래의 회로를 이용하여 LED를 X0, X1의 상태에 따라 동작시킨다.

X1	X0	LED Y0~Y7의 상태
0	0	LED 모두 OFF
0	1	LED Y0 → Y7의 순서대로 ON
1	0	LED Y0 ← Y7의 순서대로 ON
1	1	LED Y0 → Y7의 순서대로 ON



- 프로그램 : led1.c

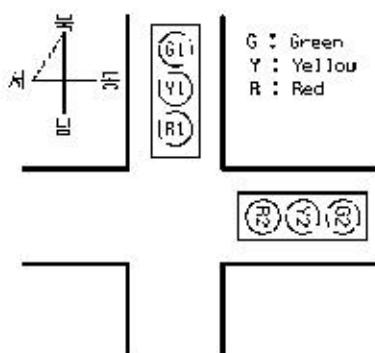
```
#include <io16f877.h>
unsigned char temp; // LED ON/OFF 저장 전역 변수
// 일정 시간 지연 함수
void delay (unsigned int i)
{
    while (i--);
}
// 시프트 레프트
void SHIFTL (void)
{
    if(temp == 0xff) // 이 동작이 처음인지 검사
    { temp = 0xfe; // 초기값 저장
        return; } // 리턴
    else {
        temp = (temp << 1)|0x01; // LED ON 값 업데이트
        if (temp == 0xff) temp = 0xfe; // 마지막 LED가 ON인 경우 초기값 저장
    }
    delay (60000); // 일정 시간 지연
}
// 시프트 라이트
```

```

void SHIFTR (void)
{
    if (temp == 0xff) // 이 동작이 처음인지 검사
    { temp = 0x7f; // 초기값 저장
        return; // 리턴
    }
    else{
        temp = (temp >> 1)|0x80; // LED ON 값 업데이트
        if (temp == 0xff) temp = 0x7f; // 마지막 LED가 ON인 경우 초기값 저장
    }
    delay (60000); // 일정 시간 지연
}
// 메인
void main(void)
{
    TRISB = 0x00; // B 포트 출력
    TRISA = 0xff; // A 포트 입력
    ADCON1 = 0x06; // A 포트 디지털 입력
    temp = 0xff; // LED ON/OFF 초기값
    do{
        PORTB = temp; // PORTB로 출력
        if (RA0) SHIFTL(); // X0 스위치="1"인 경우
        else if (RA1) SHIFTR(); // X1 스위치="1"인 경우
        else temp = 0xff; // X0, X1 스위치="1"인 경우
    }while(1); // 무한 루프
}

```

2. 교통 신호등 만들기



Y0~Y7과의 관계는
 $G1 \rightarrow Y0, Y1 \rightarrow Y1, R1 \rightarrow Y2,$
 $G2 \rightarrow Y4, Y2 \rightarrow Y5, R2 \rightarrow Y6$

4거리의 교통 신호등

● 교통 신호등 동작

다음 (1)~(7)과 같이 교통 신호등이 되도록 한다.

- (1) G1, R2만 4초 동안 점등
- (2) R2는 점등상태에서 G1만 0.1초 간격으로 4번 ON/OFF
- (3) Y1, R2만 2초 동안 점등
- (4) R1, G2만 4초 동안 점등
- (5) R1은 점등 상태에서 G2만 0.5초 간격으로 4번 ON/OFF
- (6) Y2, R1만 2초 동안 점등
- (7) (1)로 돌아간다.

● 0.5초, 2초, 4초 지연은 소프트웨어 타이머를 쓴다.

AT89S51

● 프로그램 : TRAFFIC.C

```
*****
* MDA-MULTI MICOM
* MCU : MCS51
* COMPILER : IAR
* FILE NAME : TRAFFIC.C
*****
```

```
#include <i89s51.h>      // I/O 가 정의되어 있는 헤더 파일
// 일정 시간 지연 함수
void delay (unsigned int i)
{
    while (i--);
}
// 메인
```

```

void main(void)
{
    unsigned char i;
    do{
        P1 = 0xbe;           // G1, R2 LED ON
        for ( i=0; i <4 : i++ ){ // 4초 지연
            delay(65000);
        }
        // G1 LED 4번 ON/OFF
        for ( i=0; i <4 : i++ ){
            P1 = 0xbf;           // G1 OFF, R2 ON
            delay(32500);        // 0.5초 지연
            P1 = 0xbe;           // G1, R2 LED ON
            delay(32500);        // 0.5초 지연
        }
        // Y1, R2 LED ON
        P1 = 0xbd;           // Y1, R2 LED ON
        delay(65000);        // 2초 지연
        delay(65000);

        // R1, G2 LED ON
        P1 = 0xeb;
        for ( i=0; i <4 : i++ ){ // 4초 지연
            delay(65000);
        }
        // G2 LED 4번 ON/OFF
        for ( i=0; i <4 : i++ ){
            P1 = 0xfb;           // G2 OFF, R1 ON
            delay(32500);        // 0.5초 지연
            P1 = 0xeb;           // G2, R1 LED ON
            delay(32500);        // 0.5초 지연
        }
        // Y2, R1 LED ON
        P1 = 0xdb;           // Y2, R1 LED ON
        delay(65000);        // 2초 지연
        delay(65000);
    }while(1);           // 무한 루프
}

```

2 AVR

- 프로그램 : TRAFFIC.C

```
*****  
* MDA-MULTI MICOM AVR  
* MCU : AVR  
* COMPILER : IAR  
* FILE NAME : TRAFFIC.C  
*****  
  
#include "io8535.h"  
  
unsigned char led;      // LED ON/OFF 저장 전역 변수  
  
void delay( unsigned int i ) // 일정 시간 지연 함수  
{  
    while(i--);  
}  
  
void main()  
{  
    unsigned char i;  
    DDRB=0xff; //포트 B 출력  
    do{  
        PORTB=0xbe;  
        for(i=0; i<4; i++){  
            delay(60000); //G1, R2 4초 동안 점등  
        }  
  
        for(i=0; i<4; i++){  
            PORTB=0xbff;  
            delay(40000); //G1-off R2-on 0.5초 간격  
            PORTB=0xbe;  
            delay(40000); //G1,R2 0.5초 동안 점등  
        }  
  
        PORTB=0xbd;  
        for(i=0; i<2; i++){  
            delay(60000); //Y1, R2 2초 동안 점등  
        }  
    }  
}
```

```

PORTB=0xeb;
for(i=0; i<4; i++){
delay(60000);      //R1, G2 4초 동안 점등
}

for(i=0; i<4; i++){
PORTB=0xfb;
delay(40000);      //G2-off R1-on 0.5초 간격
PORTB=0xeb;
delay(40000);      //G2,R2 0.5초 동안 점등
}

PORTB=0xdb;
for(i=0; i<2; i++){
delay(60000);      //Y2, R1 2초 동안 점등
}
}while(1); //무한 루프
}

```

3 PIC

- 프로그램 : TRAFFIC.C

```

#include <i016f877.h>      // I/O 가 정의되어 있는 헤더 파일
// 일정 시간 지연 함수
void delay (unsigned int i)
{
    while (i--);
}
// 메인
void main(void)
{
    unsigned char i;
    TRISB = 0x00; // B 포트 출력
    do{
        PORTB = 0xbe;          // G1, R2 LED ON
        for ( i=0; i <4 : i++ ){ // 4초 지연
            delay(65000);
        }
    }
}

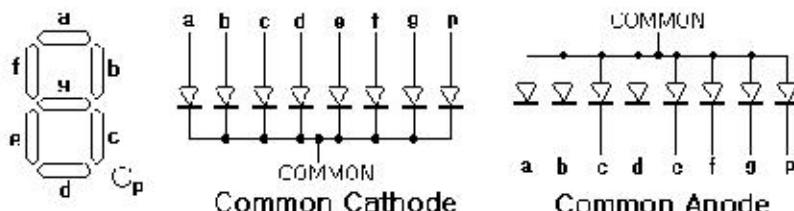
```

```
// G1 LED 4번 ON/OFF
for ( i=0; i <4 : i++){
PORTB = 0xbff;           // G1 OFF, R2 ON
delay(32500);           // 0.5초 지연
PORTB = 0xbe;            // G1, R2 LED ON
delay(32500);}           // 0.5초 지연
// Y1, R2 LED ON
PORTB = 0xbd;             // Y1, R2 LED ON
delay(65000);           // 2초 지연
delay(65000);
// R1, G2 LED ON
PORTB = 0xeb;
for ( i=0; i <4 : i++){ // 4초 지연
delay(65000);}
// G2 LED 4번 ON/OFF
for ( i=0; i <4 : i++){
PORTB = 0xfb;           // G2 OFF, R1 ON
delay(32500);           // 0.5초 지연
PORTB = 0xeb;            // G2, R1 LED ON
delay(32500);}           // 0.5초 지연
// Y2, R1 LED ON
PORTB = 0xdb;             // Y2, R1 LED ON
delay(65000);           // 2초 지연
delay(65000);
}while(1);           // 무한 루프
}
```

3. 7세그먼트 동작시키기

- 7세그먼트 구조

디스플레이 장치로 많이 사용되고 있는 7세스먼트는 OC(Common Cathode)형, CA(Common Anode)형이 있다.

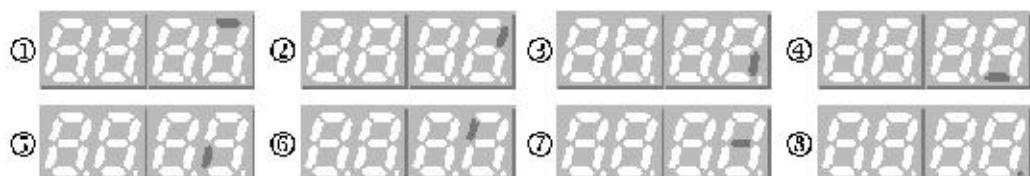


7세그먼트의 외부 모양과 내부 회로

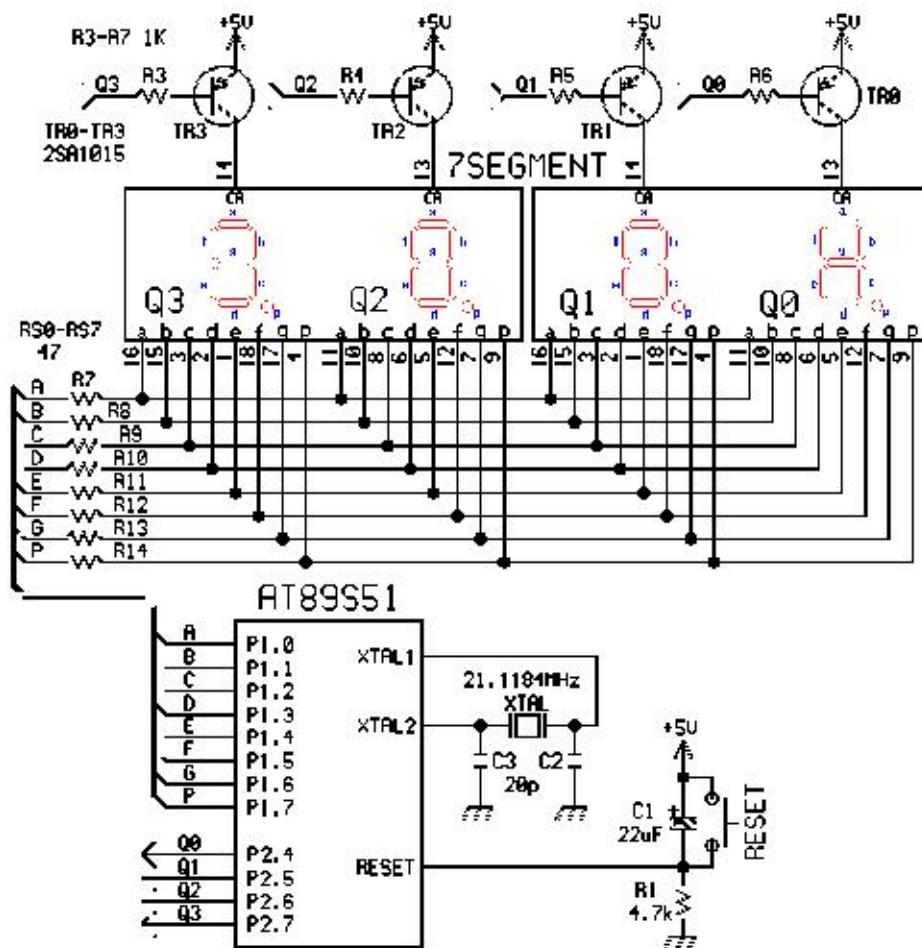
- (1) CA형은 글자를 만드는 세그먼트(a, b, ..., g)에는 "0", "common" 단자에는 "1"을 주면 원하는 숫자를 디스플레이하게 된다. 예를 들어 a=0, b=0, c=1, d=0, e=0, f=1, g=0, dp=1, common=1을 주면, 숫자 "2"가 디스플레이 된다.
- (2) OC형은 CA형과는 반대로, 글자를 만드는 세그먼트(a, b, ..., g)에는 "1", "common" 단자에는 "0"을 주면 원하는 숫자를 디스플레이하게 된다. 예를 들어 a=1, b=1, c=0, d=1, e=1, f=0, g=1, dp=0, common=1을 주면, 숫자 "2"가 디스플레이 된다.

AT89S51

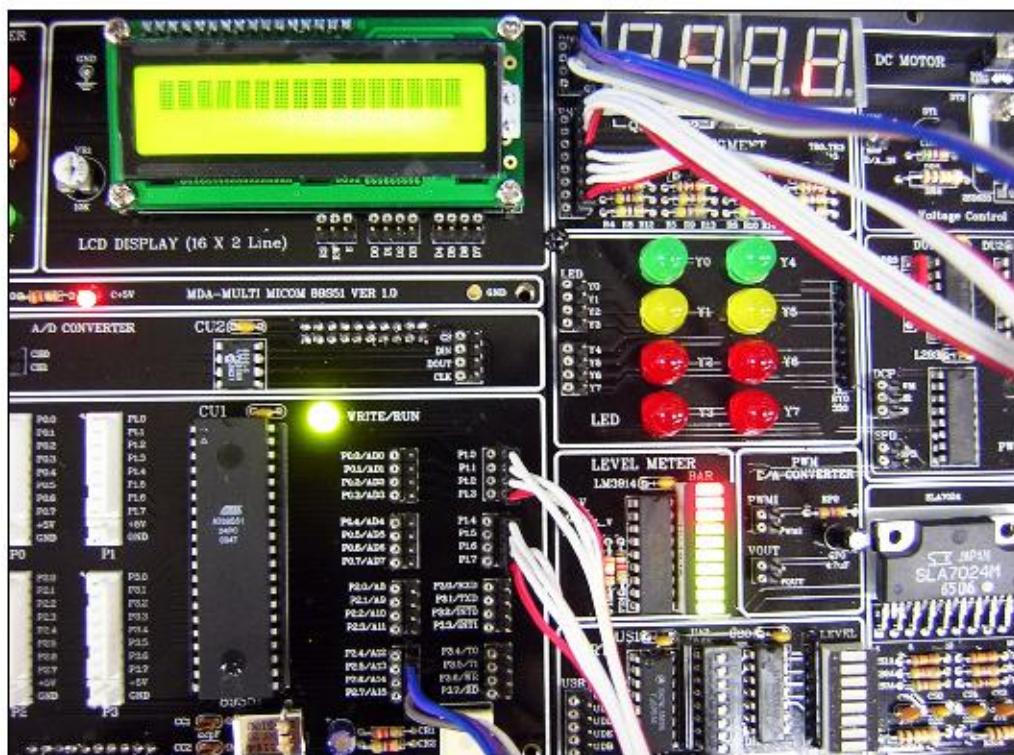
- 아래와 같이 ①→② →...→ ⑧ →①의 순서대로 동작하도록 프로그램을 만든다.



Q0 7세그먼트의 각 세그먼트 동작시키기



7세그먼트 동작을 연습하기 위한 회로



FND 점등 연습을 위해 연결된 회로

- 프로그램 : 7segment.c

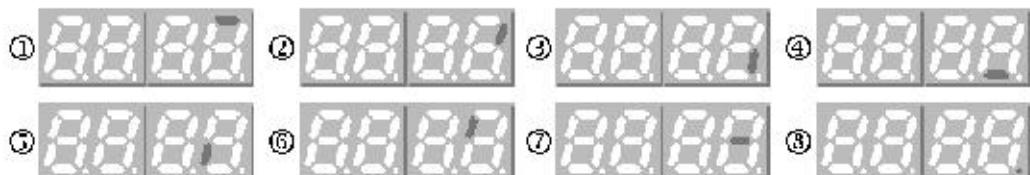
```
*****
* MDA-MULTI MICOM
* MCU : MCS51
* COMPILER : IAR
* FILE NAME : 7segment.C
*****
```

```
#include <io89s51.h> // I/O 가 정의되어 있는 헤더 파일
// 일정 시간 지연 함수
void delay (unsigned int i)
{
    while (i--);
}
// 메인
```

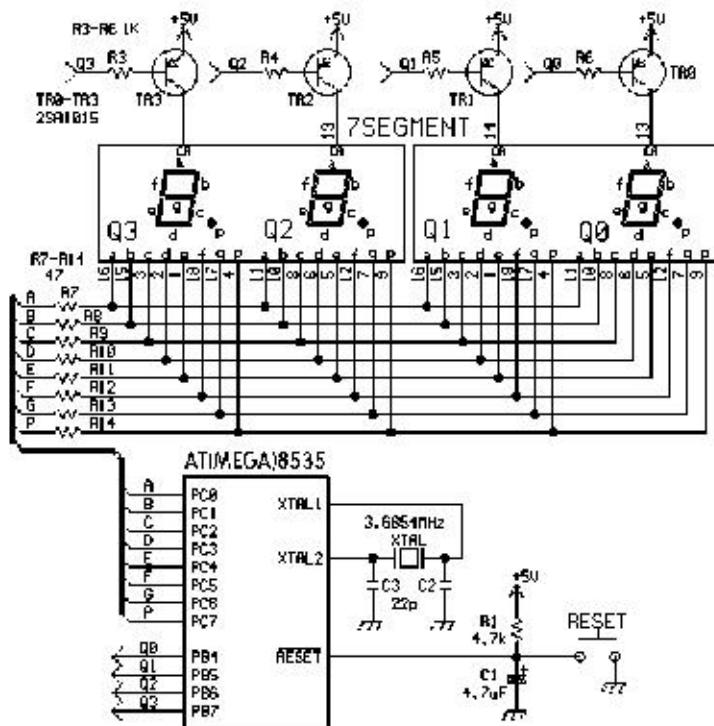
```
void main(void)
{
    unsigned char temp;
    temp = 0xfe;           // LED ON/OFF 초기 값
    P2 = 0xef;             // Q0 7세그먼트 선택
    do{
        P1 = temp;          // PORT1으로 출력
        delay( 65000 );
        temp = ( temp << 1 ) | 0x01;           // 세그먼트 ON 업데이트
        if( ( temp & 0xff ) == 0xff ) temp = 0xfe; // 마지막 세그먼트이면 초기값 저장
    }while(1);           // 무한 루프
}
```

2 AVR

- 아래와 같이 ①→② →...→ ⑧ →①의 순서대로 동작하도록 프로그램을 만든다.



Q0 7세그먼트의 각 세그먼트 동작시키기



포트의 출력 기능을 연습하기 위한 회로도

● 프로그램 : 7segment.c

```
*****
* MDA-MULTI MICOM AVR
* MCU : AVR
* COMPILER : IAR
* FILE NAME : 7segment.C
*****
```

```
#include "io8535.h"
```

```
unsigned char fnd;
```

```
void delay (unsigned int i)
{
```

```
    while(i--);
}
void main()
{
    DDRB=0xff; //포트 B 출력포트 지정
    DDRC=0xff; //포트 C 출력포트 지정
    PORTB=0xef; //포트 B에 0xef출력

    ind=0xfe; //상수 ind에 초기값 0xfe값 지정

    do{
        PORTC=ind; //포트 c에 상수 ind의 값을 넣는다.
        if(ind==0xff) ind=0xfe; //만약 ind값이 0xff와 같으면 ind에 0xfe값을 넣는다.
        delay(65000);
        ind=(ind<<1)|0x01;
            //ind값을 한비트 왼쪽으로 쉬프트 한후 0x01과 or연산을 한다.
            //즉 1111 1100 or 0000 0001= 1111 1101
        if(ind==0xff) ind=0xfe; //연산 작업후 ind의 값이 0xff가 되면
                                //다시 ind에 0xfe를 넣는다.
    }while(1);
}
```

4. 7세그먼트 글자 디스플레이하기

AT89S51

- "HELP" 가 일정시간 간격으로 디스플레이 되도록 한다.

- 프로그램 : help.c

```
*****  
* MDA-MULTI MICOM  
* MCU : MCS51  
* COMPILER : IAR  
* FILE NAME : HELP.C  
*****  
  
#include <i089s51.h>          // I/O가 정의 되어있는 헤더 파일  
code unsigned char SEGMENT[4] = {0x8c,0xc7,0x86,0x89}; // PLEH  
code unsigned char BLANK[4] = {0xff,0xff,0xff,0xff}; // 블랭크  
// 일정 시간 지연 함수  
void delay (unsigned int i)  
{  
    while (i--);  
}  
// 메인  
void main(void)  
{  
    unsigned char temp,i,j; // 임시 저장 전역 변수  
    // 루프  
    do{  
        // "HELP" 디스플레이  
        for (j=0; j<255 ; j++ ){  
            temp = 0xef;           // 세그먼트 ON 초기 값  
            for (i=0; i<4 ; i++ ){  
                P1 = SEGMENT[i];      // 7세그먼트 데이터 출력  
                P2 = temp;             // 7세그먼트 ON  
                temp = (temp << 1) | 0x01; // 다음 세그먼트 ON 업데이트  
                delay(255);           // 일정 시간 ON  
            }  
        }  
    }  
}
```

```
        }
        // "    디스플레이
        for (j=0; j<255 ; j++){
            temp = 0xef;           // 세그먼트 ON 초기 값
            for (i=0; i<4 ; i++){
                P1 = BLANK[i];      // 7세그먼트 데이터 출력
                P2 = temp;          // 7세그먼트 ON
                temp = (temp << 1) | 0x01; // 다음 세그먼트 ON 업데이트
                delay(255);         // 일정 시간 ON
            }
        }
    }while(1);
}
```

2 AVR

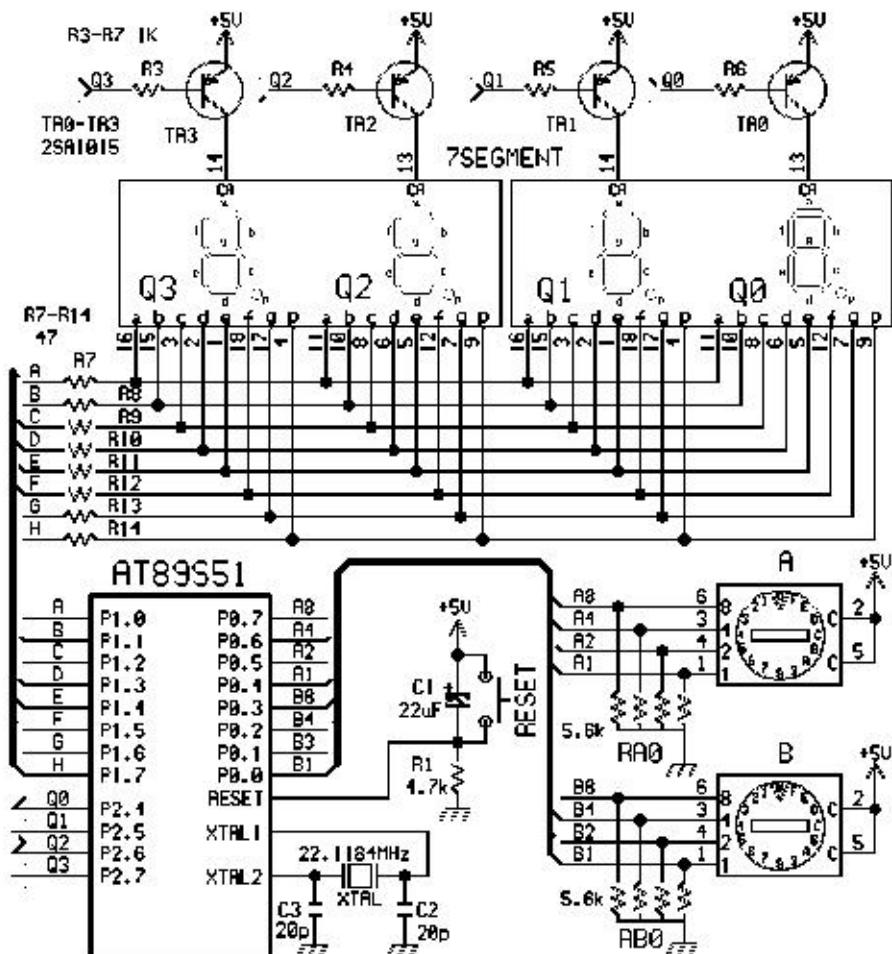
- "HELP" 가 일정시간 간격으로 디스플레이 되도록 한다.
- 프로그램 : help.c

```
#include "io8535.h"      //I/O가 정의 되어 있는 헤더 파일
_flash unsigned char SEGMENT[4] = {0x8c,0xc7,0x86,0x89}; // HELP
_flash unsigned char BLANK[4] = {0xff,0xff,0xff,0xff};   // 블랭크
// 일정 시간 지연 함수
void delay (unsigned int i)
{
    while (i--);
}
// 메인
void main(void)
{
    unsigned char temp,i,j; // 임시 저장 전역 변수
    DDRC = DDRB = 0xff; // C, B 포트 출력
    do{ // 루프
        // "HELP" 디스플레이
        for (j=0; j<255 ; j++){
            temp = 0xef;           // 세그먼트 ON 초기 값
```

```
for (i=0: i<4 : i++){  
    PORTC = SEGMENT[i];      // 7세그먼트 데이터 출력  
    PORTB = temp;            // 7세그먼트 ON  
    temp = (temp << 1) | 0x01; // 다음 세그먼트 ON 업데이트  
    delay(255);              // 일정 시간 ON  
}  
}  
// "    디스플레이  
for (j=0: j<255 : j++){  
    temp = 0xef;             // 세그먼트 ON 초기 값  
    for (i=0: i<4 : i++){  
        PORTC = BLANK[i];    // 7세그먼트 데이터 출력  
        PORTB = temp;         // 7세그먼트 ON  
        temp = (temp << 1) | 0x01; // 다음 세그먼트 ON 업데이트  
        delay(255);           // 일정 시간 ON  
    }  
}  
}  
}while(1);  
}
```

5. DIP 로터리 스위치 값 7세그먼트에 디스플레이하기

AT89S51



로터리 스위치 입력을 실험하기 위한 회로도

- DIP 로터리 스위치(Binary Coded/DIP rotary switch) 값을 읽어서 7세그먼트로 디스플레이 한다.

● 프로그램 : rotary.c

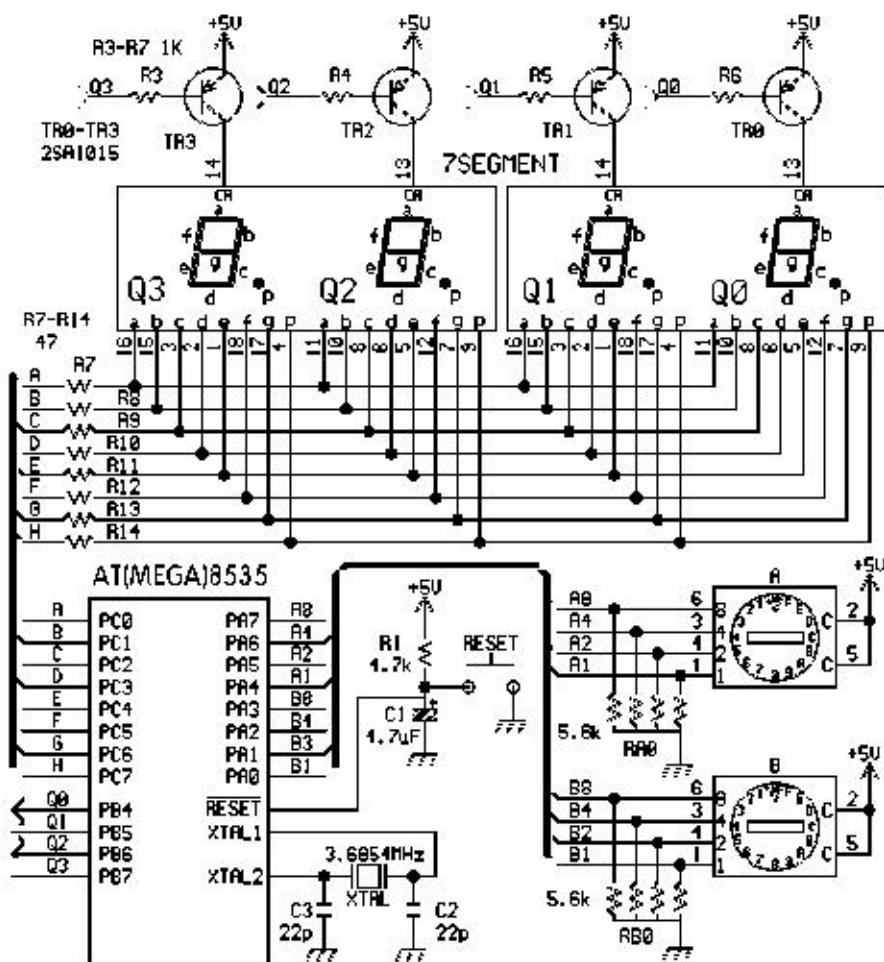
```
*****
* MDA-MULTI MICOM
* MCU : MCS51
* COMPILER : IAR
* FILE NAME : ROTARY.C
*****
```

```
#include <i089s51.h>      // I/O가 정의 되어있는 헤더 파일
// "0 - F" 숫자의 디스플레이 포맷
code unsigned char SEG[16] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,
                               0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e};

// 일정 시간 지연 함수
void delay (unsigned char i)
{
    while (i--);
}

// 메인
void main(void)
{
    unsigned char temp,temp1;          // 임시 저장 변수
    // 루프
    do{
        temp1 = 0xef;
        temp = P0;                    // 입력
        // 하위 디지트 출력
        P1 = SEG[temp & 0x0f];       // 7세그먼트 데이터 출력
        P2 = temp1;                  // 7세그먼트 ON
        temp1 = (temp1 << 1) | 0x01; // 다음 세그먼트 ON 업데이트
        delay(500);                 // 일정 시간 ON
        // 상위 디지트 출력
        P1 = SEG[(temp >> 4)& 0x0f]; // 7세그먼트 데이터 출력
        P2 = temp1;                  // 7세그먼트 ON
        delay(500);                 // 일정 시간 ON
    }while(1);
}
```

2 AVR



로터리 스위치 입력을 실험하기 위한 회로도

- 프로그램 : rotary.c

```
*****
* MDA-MULTI MICOM AVR
* MCU : AVR
* COMPILER : IAR
* FILE NAME : ROTARY.C
*****
```

```

#include "io8635.h"

unsigned char SEGMENT[16]={ 0xc0.0xf9,0xa4.0xb0.0x99,0x92,0x82,0xf8,
                            0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e};

void delay(unsigned int i)
{
    while (i--);
}

void main(void)
{
    unsigned char k,EN;

    DDRC=0xf; //포트 C에 7세그먼트 값 출력
    DDRB=0xf; //포트 B에 7세그먼트 인에이블 출력

    DDRA=0x00; //포트 A는 로터리 스위치 입력
    do{
        EN=0xef; //EN이란 상수에 0xef의 값을 넣는다 이값은 7세그먼트가 인에이블
        될 포트의 값이다.

        k=PIN_A; //k라는 상수에 포트A의 입력핀 어드레스를 넣는다.

        PORTC = SEGMENT[(k>>4) & 0x0f];
        //포트 C에 세그먼트값을 넣는다.
        //포트 A의 입력 핀 상위 값을 표현하기 위하여 어드레스에서 상위 4비트를 오른쪽
        으로 쉬프트하여 0x0f와 연산을 한후 PORT c에 넣는다.
        //즉 1000 000->상위 4비트 오른쪽 쉬프트 -->0000 1000 & 0000 1111=0000
        1000의 값이 나온다.
        //0x0f와 &연산을 하지 않을경우 7->8로 갈때 전체적으로 on되었다가 간다.

        PORTB=EN; //포트 B에 상수 EN(0xef)의 값을 넣는다.
        //즉 7세그먼트 Q0를 on

        EN=(EN<<1)|0x01; //상수 EN값을 왼쪽으로 쉬프트 연산 하기 위하여 or 연산을
        한다.
        delay(255); //지연값을 적게 주어 항상 켜져있는것처럼 보이게 한다.

        PORTC = SEGMENT[k & 0x0f];
        //포트 C에 세그먼트값을 디스플레이 한다.
        //포트 A의 입력핀 하위값을 표현한다.

```

3부/제 1장 I/O 포트 연습

//0x0f와 &연산을 하지 않을 경우 7->8로 갈때 전체적으로 on되었다가 간다.

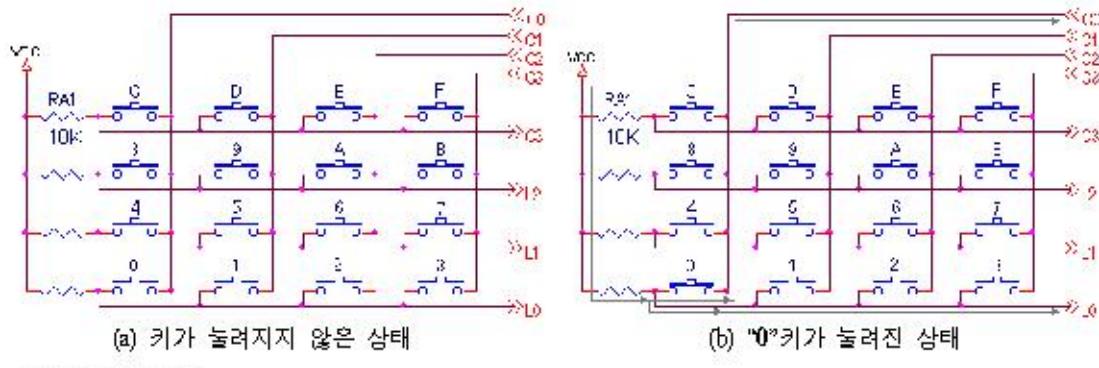
PORTE=EN: //포트 E에 EN=(EN<<1)|0x01의 연산값을 넣는다.
delay(255); //지연값을 적게 주어 항상 켜져있는것처럼 보이게 한다.

```
}while(1) :  
}
```

6. 키보드 스캔하기

● 키보드 스캔

행렬 형태의 키보드는 가장 비싸지 않은 입력 장치이며, 일반적인 키보드의 구조는 아래 그림과 같이 보통 행렬 형태로 되어 있다. 키를 눌렀을 때 한 행과 한 열이 접촉하게 되며, 4개의 열은 출력 포트, 4개의 행을 입력 포트에 연결하여야 한다.



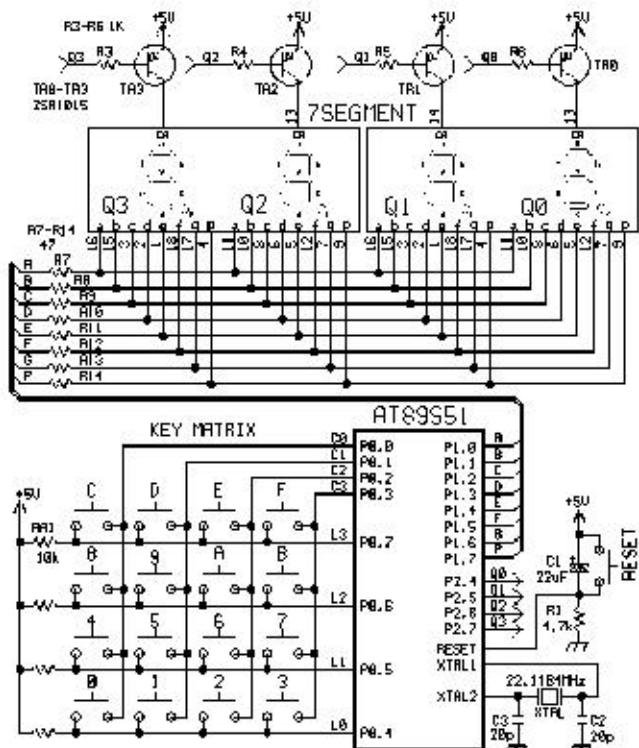
스캔 방법은 다음과 같다.

- (1) C3~C0에 "1110"을 출력하고, L3~L0을 입력한다. 키가 눌려진 것이 없으면, L3~L0은 "1111"이 입력되고, 그림 2-21(b)에서처럼 '0' 키가 눌려진 경우 "1110"이 입력된다.
- (2) C3~C0에 "1101"을 출력하고, L3~L0을 입력한다. 키가 눌려진 것이 없으면, L3~L0은 "1111"이 입력되고, 눌려진 키가 있으면 해당되는 행이 "0"이 입력된다.
- (3) C3~C0에 "1011"을 출력하고, L3~L0을 입력한다. 키가 눌려진 것이 없으면, L3~L0은 "1111"이 입력되고, 눌려진 키가 있으면 해당되는 행이 "0"이 입력된다.
- (4) C3~C0에 "0111"을 출력하고, L3~L0을 입력한다. 키가 눌려진 것이 없으면, L3~L0은 "1111"이 입력되고, 눌려진 키가 있으면 해당되는 행이 "0"이 입력된다.
- (5) (1)~(4)를 반복해서 마지막에 입력된 값을 유효한 키 검출로 인정하고, 소프트웨어로 눌려진 키를 해독하게 된다. 그림 2-21의 구조로 되어 있는 경우, 0~F 키의 검출되는 값은 아래 표와 같이 된다.

4x4 키 스캔 코드

키	C3~C0 : L3~L0	키	C3~C0 : L3~L0
0	1110 1110	8	1110 1011
1	1101 1110	9	1101 1011
2	1011 1110	A	1011 1011
3	0111 1110	B	0111 1011
4	1110 1101	C	1110 0111
5	1101 1101	D	1101 0111
6	1011 1101	E	1011 0111
7	0111 1101	F	0111 0111

1 AT89S51



4x4 키 매트릭스 회로도

● 프로그램 : key.c

```
*****
* MDA-MULTI MICOM
* MCU : MCS51
* COMPILER : IAR
* FILE NAME : KEY.C
*****
동작 설명
KEY MATRIX의 0.4.8.C 키를 누르면
7-SEGMENT의 Q0에 디스플레이 된다.
*****
```

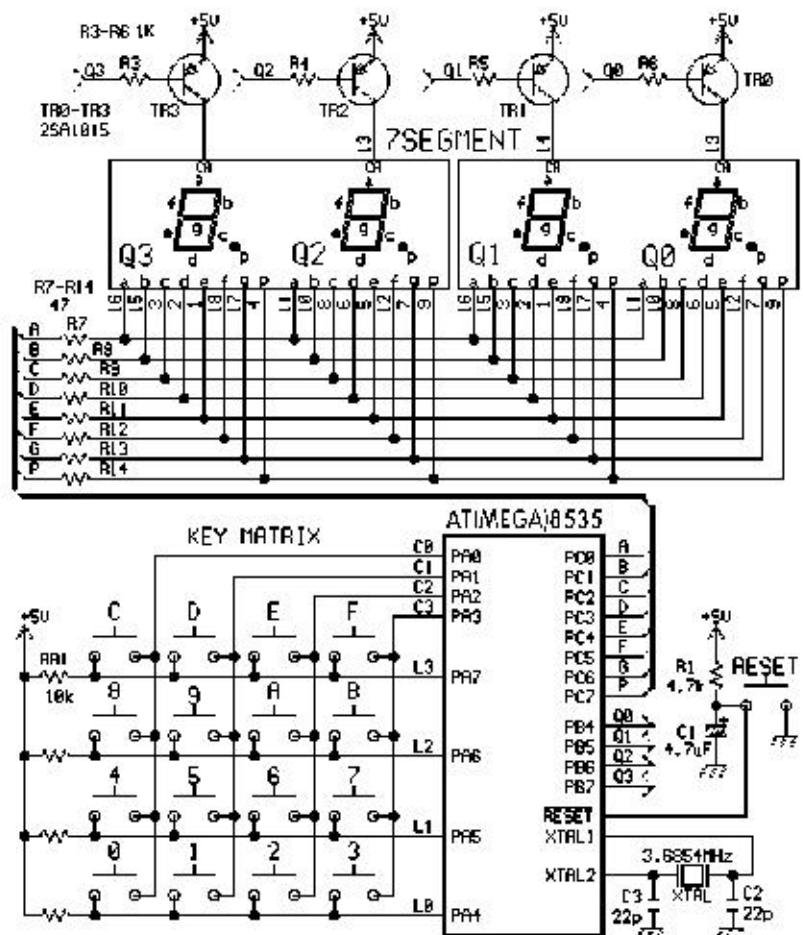
```
#include <io89s51.h>      // I/O가 정의되어 있는 헤더 파일
unsigned char KEY;        // 키 저장 변수
unsigned char KEY2;
unsigned char FLAG;       // 키 검출 변수
// "0 - F" 숫자의 디스플레이 포맷
code unsigned char SEG[16] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,
                               0x80,0x90,0x88,0x88,0xc6,0xa1,0x86,0x8e};
// 0-F키 내부 코드
code unsigned char KCODE[16] = {0x00,0x04,0x08,0x0C,0x01,0x05,0x09,0x0d,
                                0x02,0x06,0x0a,0x0e,0x03,0x07,0x0b,0x0f};
// 키보드 스캔 및 7세그먼트 디스플레이
void SCAN1(void)
{
    unsigned char i,temp,key1;
    KEY = key1 = 0;           // 변수 클리어
    FLAG = 1;                // 플래그 클리어
    P0 = 0xfe;               // 키보드 열 출력
    temp = P0;                // 키보드 행 입력
    temp = (temp >> 4) | 0xf0; // 하위 4비트로 이동
    for (i=0; i<4; i++){     // 4번 루프
        if (!(temp & 0x01)){ // 최하위 비트 "0"을 검사
            key1 = KEY; FLAG = 0; // 키 위치 코드 저장
        }
        temp >>= 1;           // 입력된 키 값을 1비트 이동
        KEY++;                // 위치 코드 +1
    } // i FOR
    // 위치 코드 값 저장
    KEY = key1 & 0x0f;
```

```
}

// 키보드를 스캔한다.
void SCAN(void)
{
    unsigned char i;
    for (i=0; i<4; i++){ // 바운스 현상을 제거하기 위하여 4번 루프
        do{                      // 키보드 스캔 및 세그먼트 디스플레이
            SCAN1();
        }while(FLAG == 0);
    }
    // 키가 눌려질 때까지 스캔
    do{
        SCAN1();
    }while(!(FLAG == 0));
    // 평선 키일 경우는 의미가 없다
    KEY2 = KCODE[KEY]; // 키 값 코드 변환
}

// 메인
void main(void)
{
    P2 = P1 = 0xff; // PORT 2, PORT 1 초기 설정
    do{
        SCAN();           // 키 스캔
        P1 = SEG[KEY2]; // 7세그먼트 데이터 출력
        P2 = 0xef;       // Q0 7세그먼트 ON
    }while(1);
}
```

2 AVR



4 x 4키 매트릭스 회로도

- 프로그램 : key.c

```
*****
* MDA-MULTI MICOM AVR
* MCU : AVR
* COMPILER : IAR
* FILE NAME : KEY.C
*****
*****  
동작 설명  
KEY MATRIX의 0.4.8.C 키를 누르면  
7-SEGMENT의 Q0에 디스플레이 된다.  
*****  
#include "io8535.h" // I/O가 정의 되어있는 헤더 파일  
unsigned char KEY; // 키 저장 변수  
unsigned char KEY2;  
unsigned char FLAG; // 키 검출 변수  
// "0 - F" 숫자의 디스플레이 포맷  
_flash unsigned char SEG[16] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,  
                                0x80,0x90,0x88,0x89,0xc6,0xa1,0x86,0x8e};  
// 0-F키 내부 코드  
_flash unsigned char KCODE[16] = {0x00,0x04,0x08,0x0C,0x01,0x05,0x09,0x0d,  
                                0x02,0x06,0x0a,0x0e,0x03,0x07,0x0b,0x0f};  
// 키보드 스캔 및 7세그먼트 디스플레이  
void SCAN10()  
{  
    unsigned char i,temp,key1;  
    KEY = key1 = 0; // 변수 클리어  
    FLAG = 1; // 플래그 클리어  
    PORTA = 0xfe; // 키보드 열 출력  
    asm ("nop"); // 입력을 위한 지연  
    asm ("nop");  
    temp = PINA; // 키보드 행 입력  
    temp = (temp >> 4) | 0xf0; // 하위 4비트로 이동  
    for (i=0; i<4; i++){ // 4번 루프  
        if (!(temp & 0x01)){ // 최하위 비트 "0"을 검사  
            key1 = KEY; FLAG = 0; // 키 위치 코드 저장
```

```

        }
        temp >>= 1;           // 입력된 키 값을 1비트 이동
        KEY++;                // 위치 코드 +1
    } // i FOR
    // 위치 코드 값 저장
    KEY = key1 & 0x0f;
}
// 키보드를 스캔한다.
void SCAN()
{
    unsigned char i;
    for (i=0; i<4; i++){ // 바운스 현상을 제거하기 위하여 4번 루프
        do{               // 키보드 스캔 및 세그먼트 디스플레이
            SCAN1();
        }while(FLAG == 0);
    }
    // 키가 눌려질 때까지 스캔
    do{
        SCAN1();
    }while(!(FLAG == 0));
    // 평선 키일 경우는 의미가 없다
    KEY2 = KCODE[KEY]; // 키 값 코드 변환
}
// 메인
void main(void)
{
    DDRB = 0xff;      // B 포트 출력
    DDRC = 0xff;      // C 포트 출력
    DDRA = 0x0f;      // PA0-PA3 출력, PA4-PA7 입력
    PORTC = 0xff;      // C 포트 초기 설정
    PORTB = 0xff;      // B 포트 초기 설정
    do{
        SCAN();        // 키 스캔
        PORTC = SEG[KEY2]; // 7세그먼트 데이터 출력
        PORTB = 0xet; // Q0 7세그먼트 ON
    }while(1);
}

```

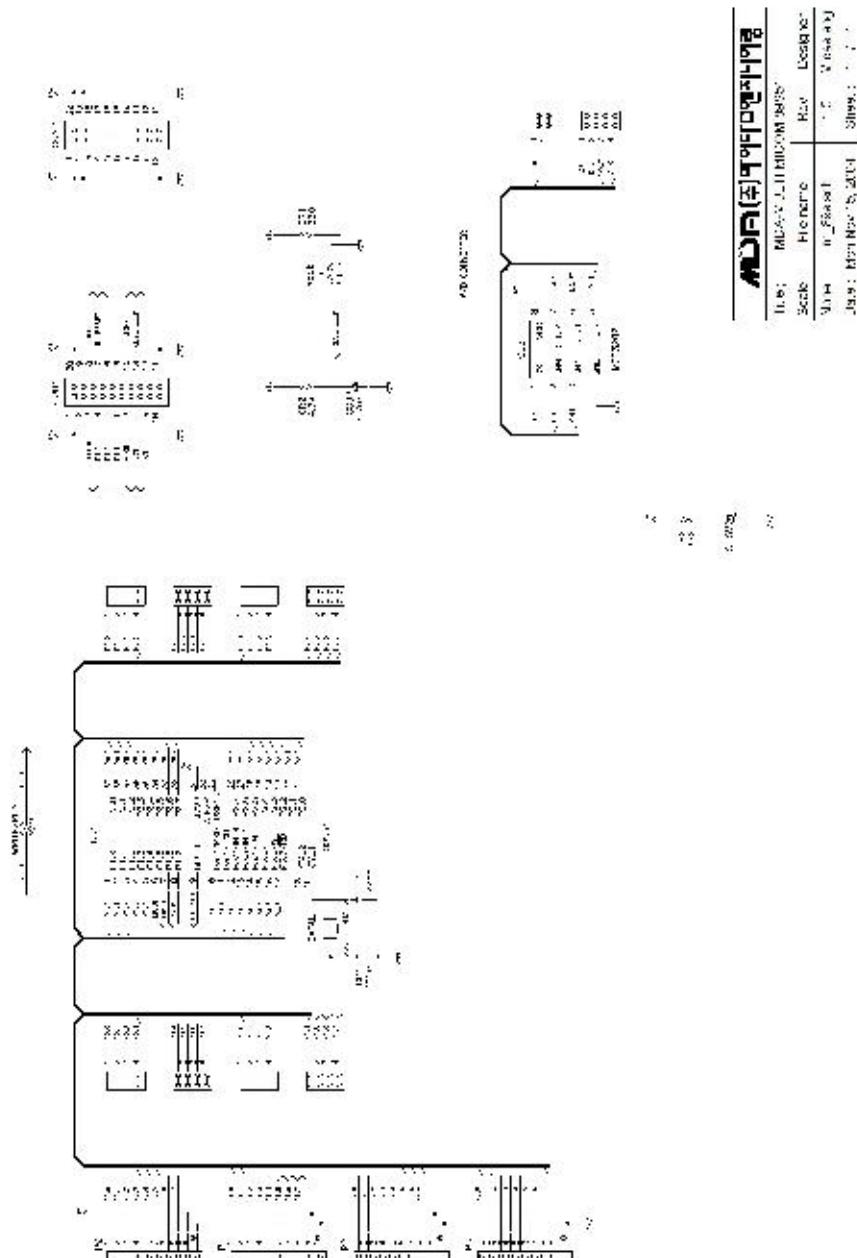



부록 MDA-Multi MICOM USB 회로도

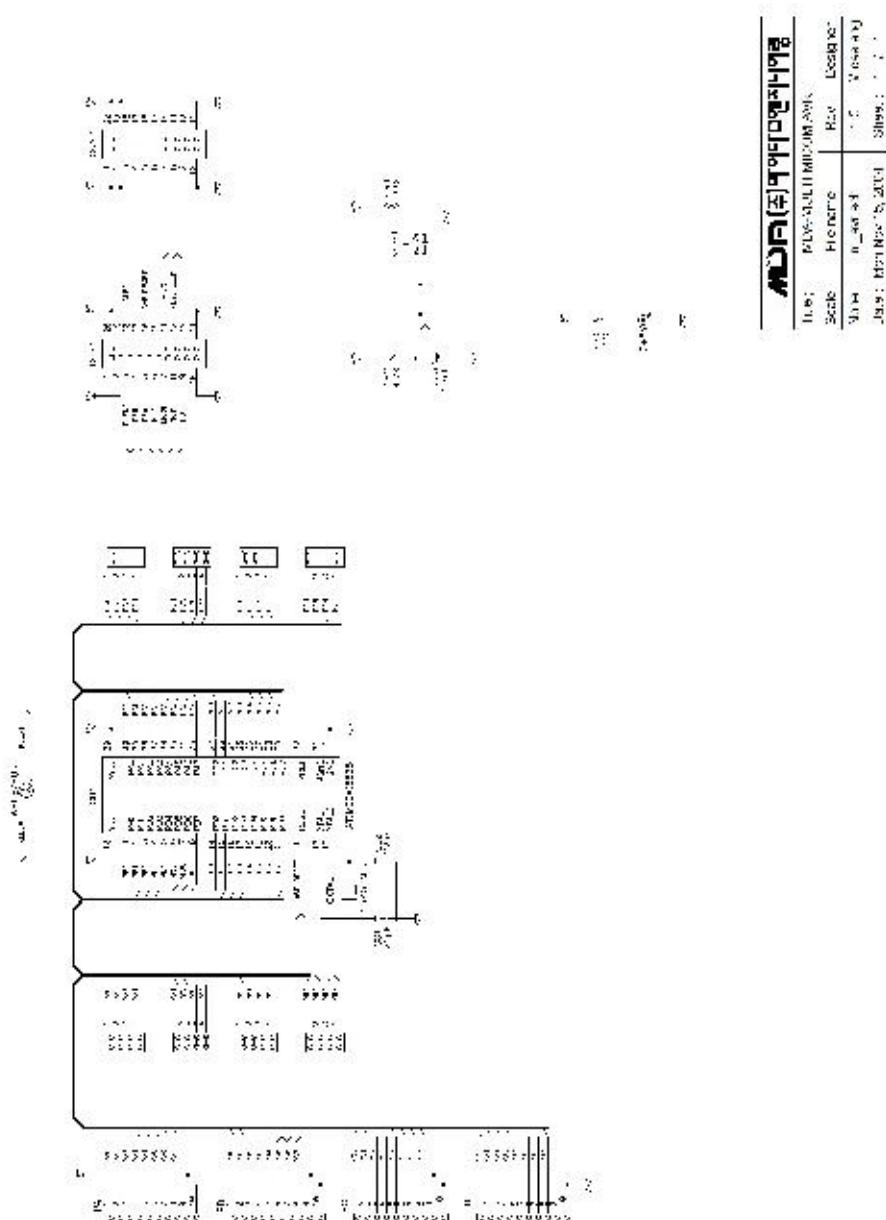
(주) 마이다스 엔지니어링에서는 사용자 편의를 위하여 각 보드의 회로도를 공개합니다.

- AT89S51 회로도
- AT(MEGA)8535 회로도
- PIC 회로도
- I/O 회로도
- I/O UP BOARD 회로도

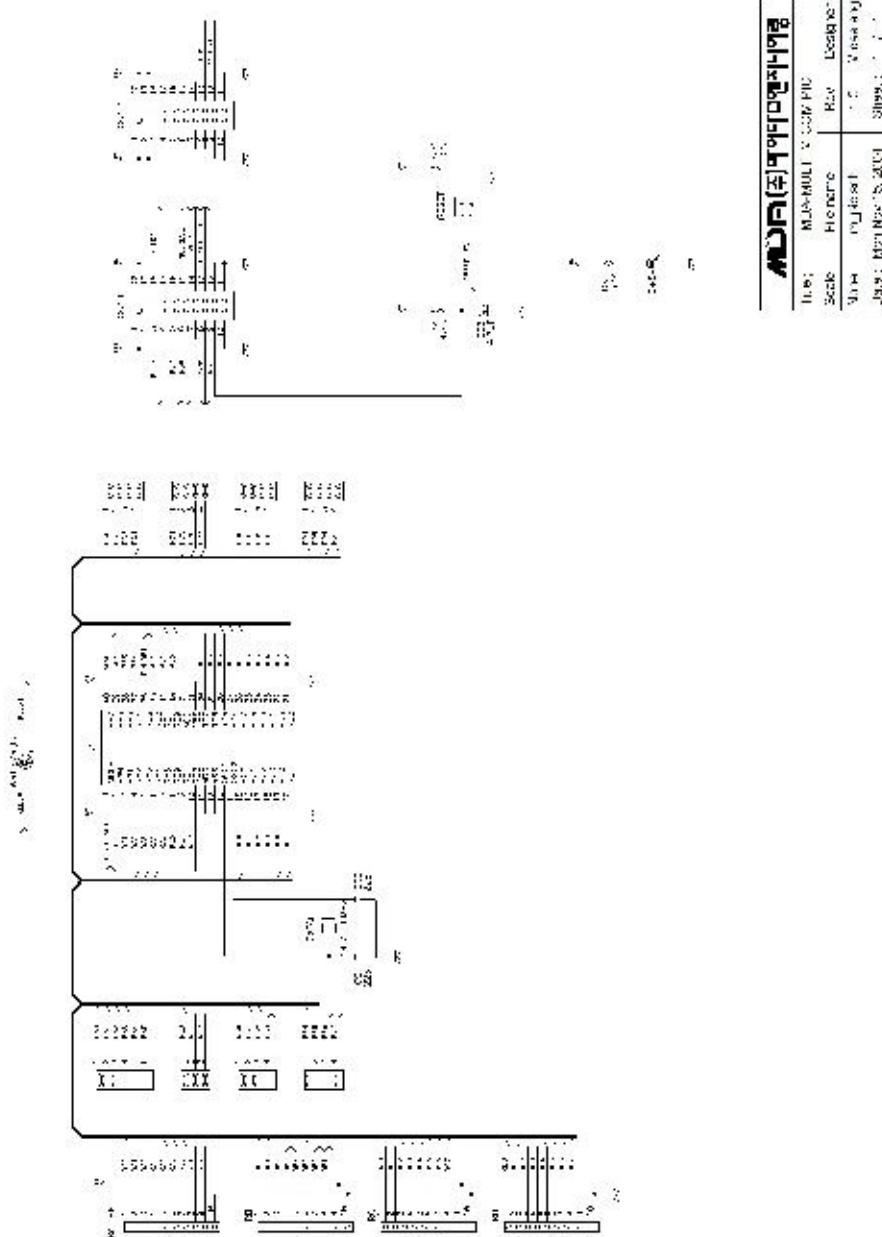
1. AT89S51 회로도



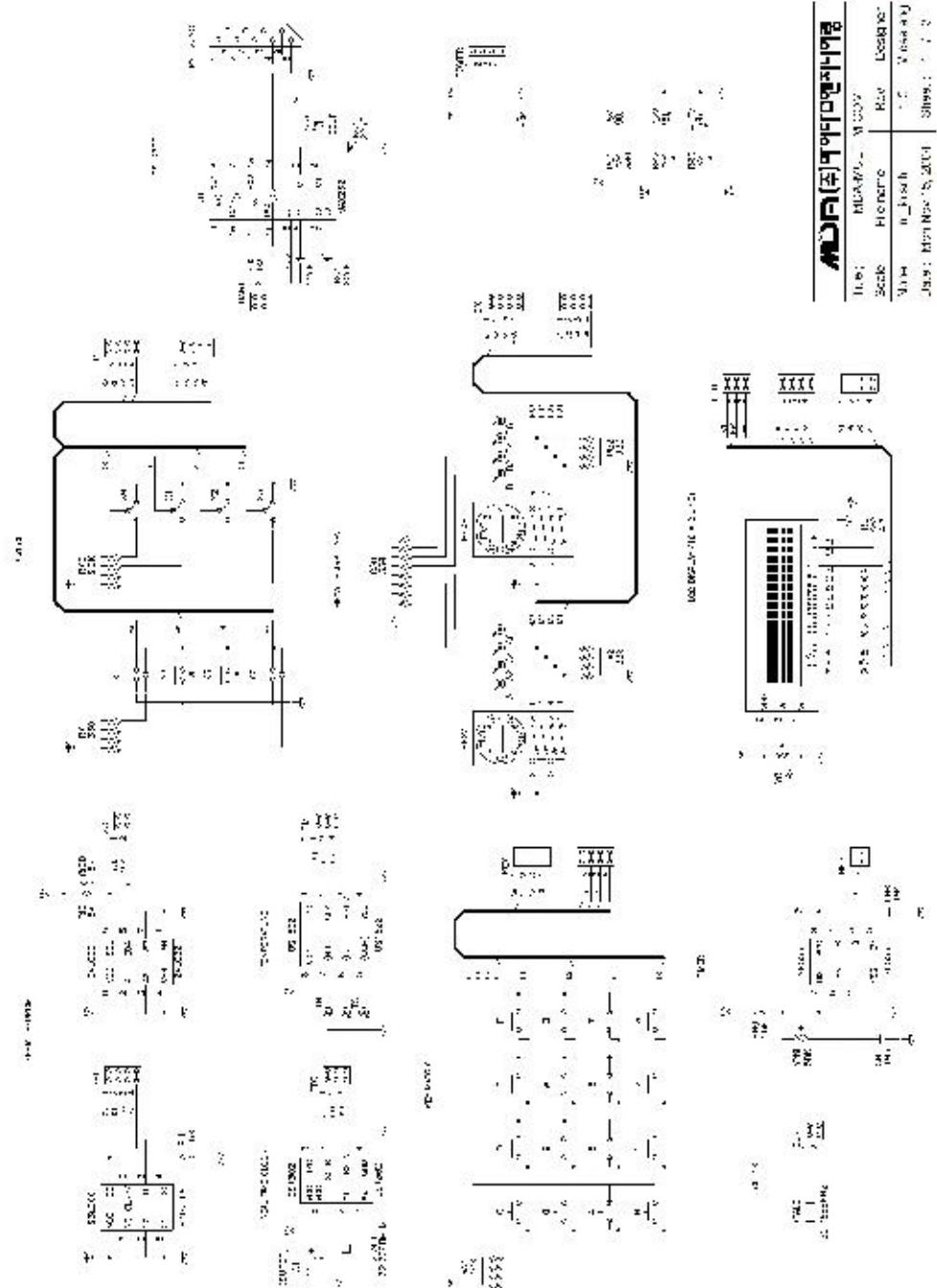
2. AVR : AT(MEGA)8535 회로도

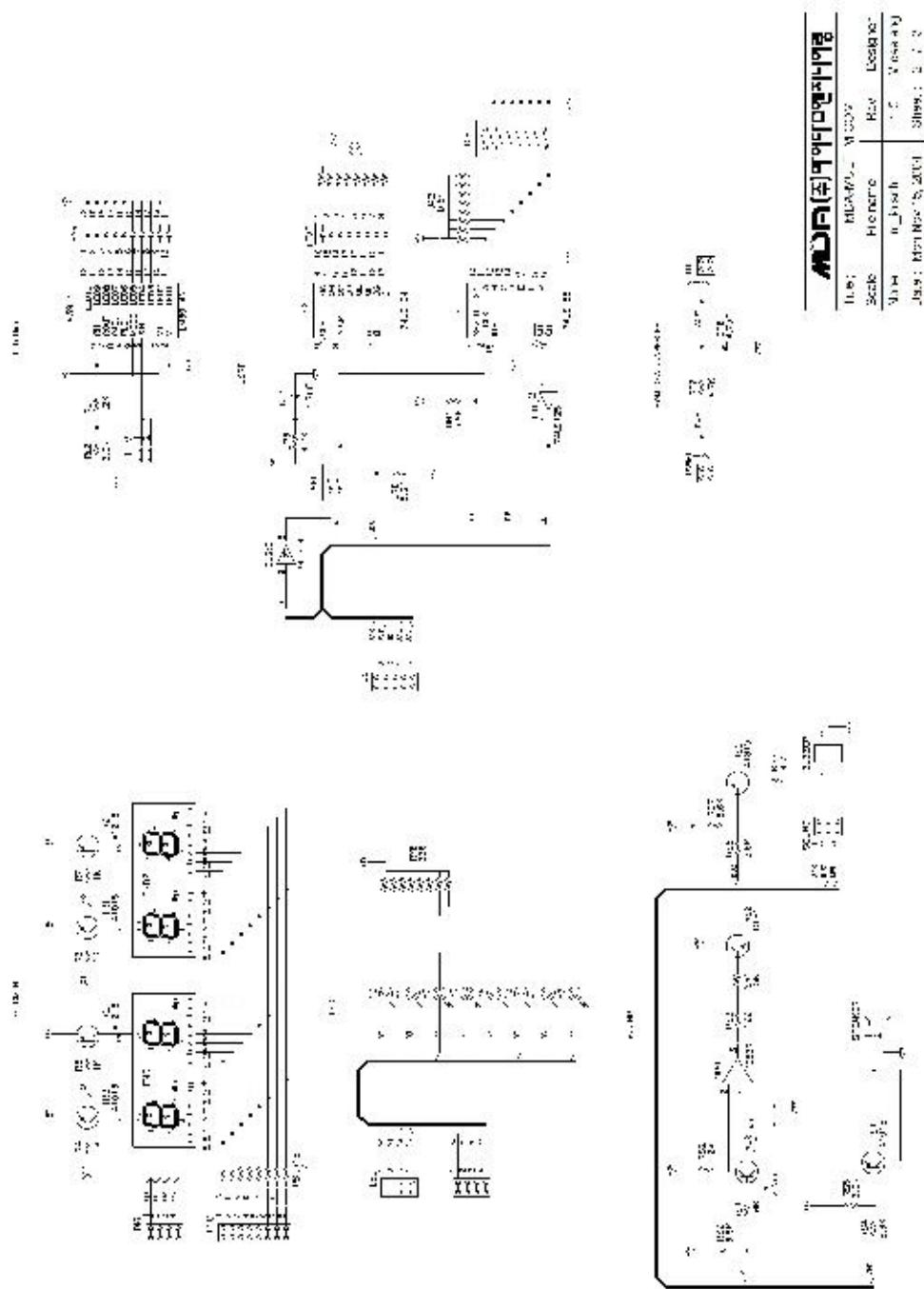


3. PIC : PIC16F874/7 회로도

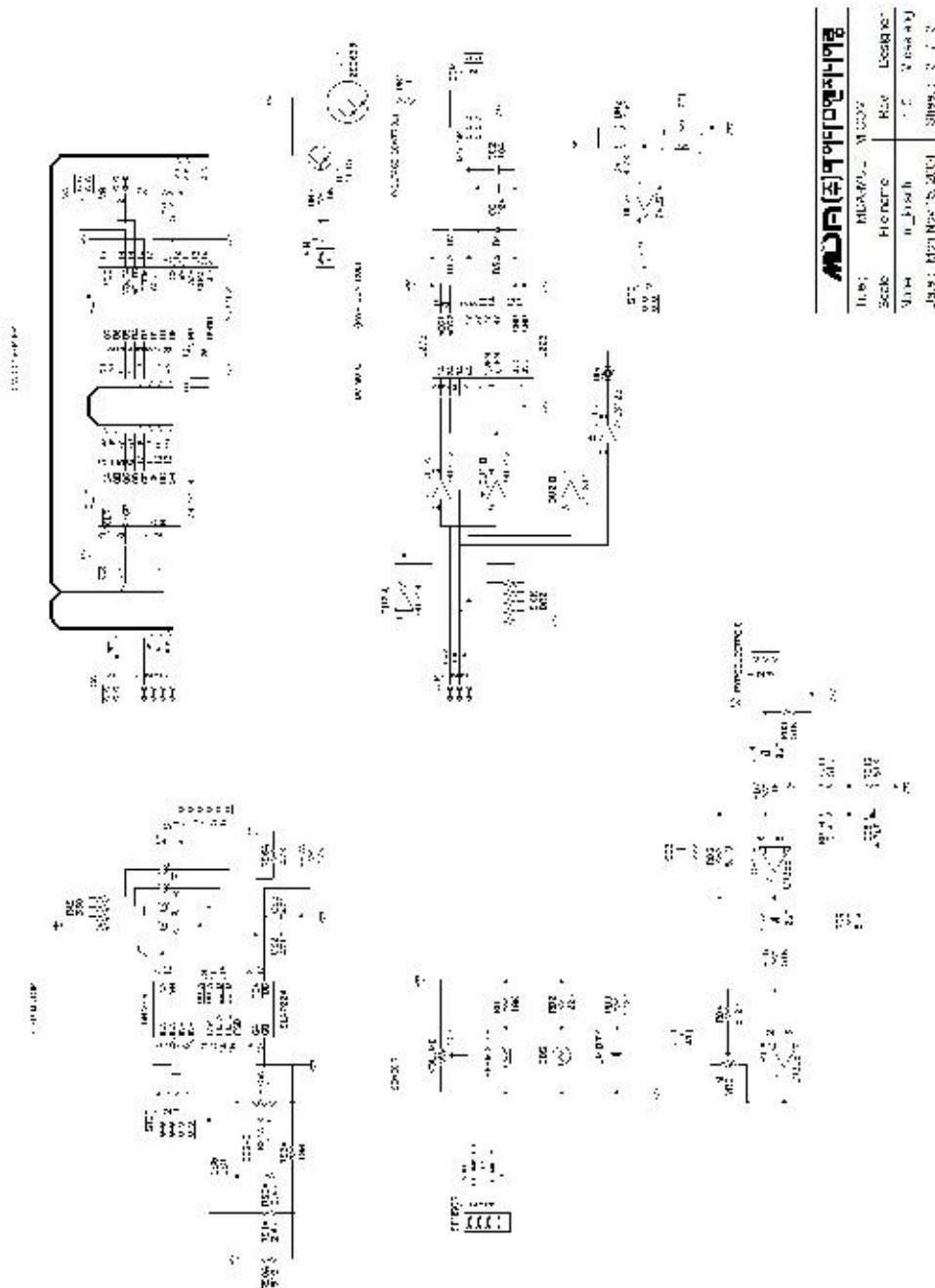


4. MDA-Multi MICOM USB I/O 회로도

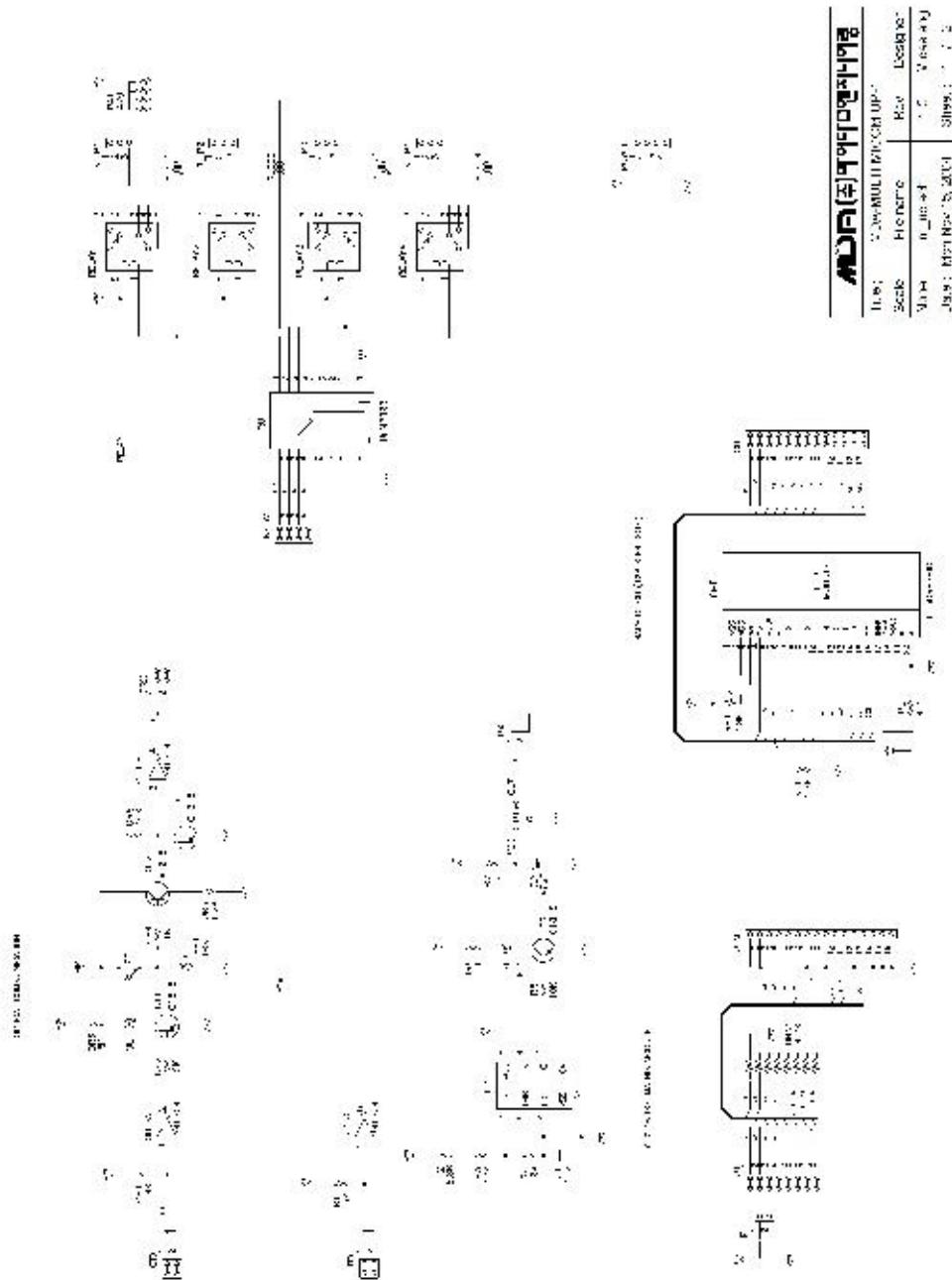




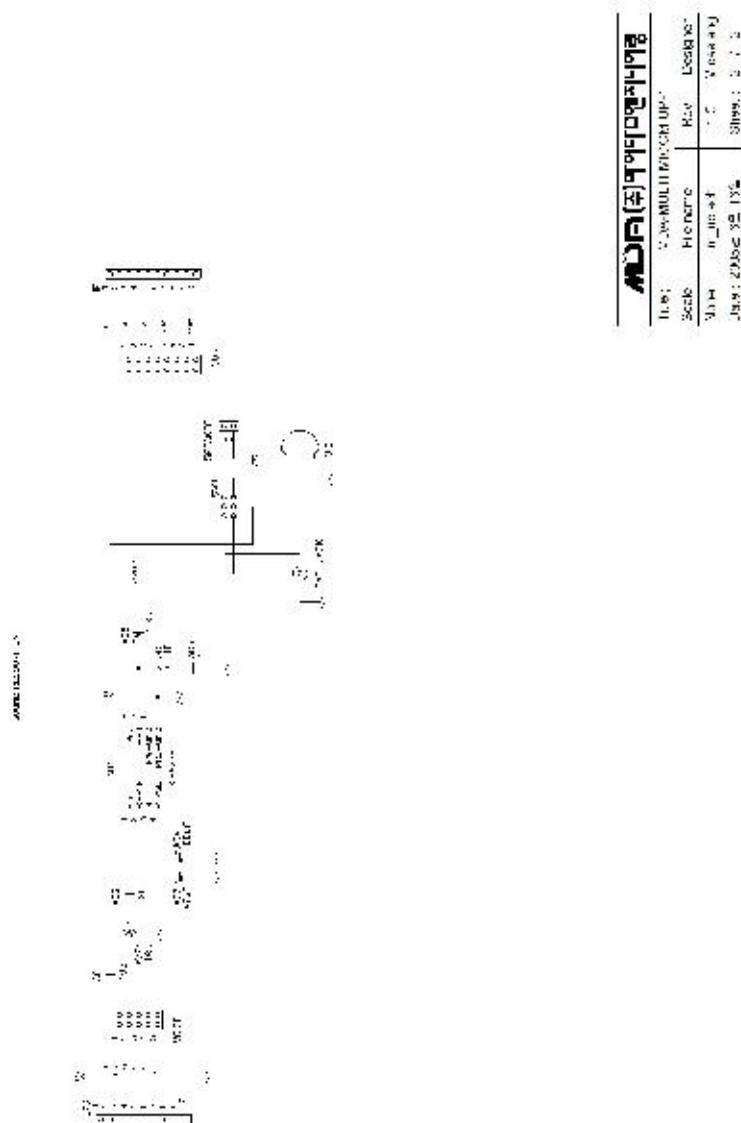
부록6. MDA-Multi MICOM USB V0 회로도



5. MDA-Multi MICOM USB UP BOARD 회로도



부록7. MDA-Multi MICOM USB UP BOARD 회로도



Copyright © 2006 Midas Engineering Co., Ltd.
Midas Engineering Co., Ltd.
User guide
Serial No. 061216 

Multi Micom USB

*편집일 : 2006-12-14

*편집자 : 이은규

*편집내용

I. 기존 Multi Micom Manual 일부 오타 수정

차례 3부분 소■→소■

1부 제목장 소■→소■

P.8 제목: 소■→소■

 머리말: 소■→소■

P.9 머리말: 소■→소■

P.15 제목: 소■→소■

 머리말: 소■→소■

P.16 머리말: 소■→소■

P.9

어셈블과 컴파일은 동시에 액티브 되지 않고 소TM 파일에 의해서 자동으로 설정된다.

→어셈블과 컴파일은 동시에 액티브 되지 않고 소스 파일에 의해서 자동으로 설정된다.

II. 글 추가

P.55 그림1-11

P.82 그림2-20

III. 뒷표지

출력본에 II번은 수정되어 있었으므로 출력은

I(차례,6,8,9~19)III번 만하였음. 06.04.17-----